PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L

A2

(11) International Publication Number: WO 98/37657

(43) International Publication Date: 27 August 1998 (27.08.98)

(21) International Application Number: PCT/US98/03093

(22) International Filing Date: 19 February 1998 (19.02.98)

(30) Priority Data:

08/802,509	20 February 1997 (20.02.97)	US
08/803,088	20 February 1997 (20.02.97)	US
08/804,784	20 February 1997 (20.02.97)	US

(71) Applicant: 3COM CORPORATION [US/US]; 8100 North McCormick Boulevard, Skokie, IL 60076 (US).

(72) Inventor: TOWNSHEND, Brent; 156 University Drive, Menlo Park, CA 94025 (US).

(74) Agent: HUGHES, A., Blair; McDonnell Boehnen Hulbert & Berghoff, 300 South Wacker Drive, Chicago, IL 60606 (US).

(81) Designated States: AU, CA, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: HIGH SPEED COMMUNICATIONS SYSTEM FOR ANALOG SUBSCRIBER CONNECTIONS

(57) Abstract

A new data communication system is described which enables data transmission over existing telephone lines at rates higher than possible with existing methods, including conventional modems. By using a novel asymmetric configuration of the two communicator endpoints, the currently-accepted maximum theoretical limits on the data rate are no longer applicable. One endpoint is connected directly to a digital telephone network, whereas the other endpoint uses a conventional telephone connection. This reduces the transmission problem to compensation for a single telephone line interface and a single analog local loop. Means of providing this compensation and the required clock synchronization were also created, enabling a practical implementation of the system. The new system can achieve rates up to 64,000 bits-per-second and has broad utility in several active areas including wide-band audio transmission, video transmission, networking, facsimile transmission, and remote computer access.

HIGH SPEED COMMUNICATIONS SYSTEM FOR ANALOG SUBSCRIBER CONNECTIONS

BACKGROUND OF THE INVENTION

The field of the present invention pertains generally to data communications equipment, and more particularly to a device for transmitting digital data over a telephone connection.

Data communication plays an important role in many aspects of today's society. Banking transactions, facsimiles, computer networks, remote database access, credit-card validation, and a plethora of other applications all rely on the ability to quickly move digital information from one point to another. The speed of this transmission directly impacts the quality of these services and, in many cases, applications are infeasible without a certain critical underlying capacity.

At the lowest levels, most of this digital data traffic is carried over the telephone system. Computers, facsimile machines, and other devices frequently communicate with each other via regular telephone connections or dedicated lines which share many of the same characteristics. In either case the data must first be converted into a form compatible with a telephone system designed primarily for voice transmission. At the receiving end the telephone signal must be converted back into a data stream. Both tasks are accomplished by modems.

A modem performs two tasks corresponding to the needs above: modulation, which converts a data stream into an audio signal that can be carried by the telephone system, and demodulation, which takes the audio signal and reconstructs the data stream. A pair of modems, one at each end of a connection, allows bidirectional communication between the two points. The constraints on the audio signal create the limitations on the speed at which data can be transferred using modems. These constraints include a limited bandwidth and degradation of data by noise and crosstalk. The telephone system typically can carry only signals that range in frequency between 300 Hz and 3,400 Hz. Signals outside this range are sharply attenuated. This range was built into the design of the telephone system since it covers a significant portion of by the human voice spectrum. However, the bandwidth of a channel is one factor that determines the maximum attainable data rate. With all other factors constant, the data rate is directly proportional to the bandwidth.

Another factor is the distortion of the audio signal or any other signal that the communications endpoints cannot control. This includes electrical pickup of other signals being carried by the telephone system (crosstalk), electrical noise, and noise introduced by conversion of the signal from one form to another. The last type will be expanded upon in later discussion.

For general utility, modems are designed to be operable over most telephone connections. Thus, they must be designed for worst-case scenarios, which include bandwidth limitations and significant noise that cannot be removed. Even so, substantial progress has been made on modem design in the past several years. Devices capable of operating at speeds up to 28,800 bits per second are now commonly available. See International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), Recommendation V.34, Geneva, Switzerland (1994) which is hereby incorporated herein by reference. However, theoretical arguments based on the channel bandwidth and noise levels show that the maximum possible speed has nearly been obtained and further significant increases are highly unlikely with the given constraints. This is discussed in C.E. Shannon, "*A Mathematical theory of Communication,*" Bell System Technical Journal 27:379-423,623 -656 (1948) which is hereby incorporated herein by reference.

Unfortunately, although speeds approaching 30,000 bits per second (or 3,600 bytes per second) make many data communications applications feasible, conventional modem transmission is still not fast enough for all uses. At these speeds, transmission of text is fast, and low-quality audio, such as digitized speech, is acceptable. However, facsimile or still-image transmission is slow, while high-quality audio is limited and full-motion video has not been satisfactorily achieved. In short, what is needed is greater data transmission capability. This is a prerequisite for the new applications and is a necessity for maximizing the performance of many existing applications.

Of course the telephone companies, cable-television providers, and others are not ignorant of these increasing data transmission needs. One approach to providing higher speed data connections to businesses and residences is to provide end-to-end digital connectivity, eliminating the need for additional modems. One offering of such a service is the Integrated Services Digital Network (ISDN). See: International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), "*Integrated Services Digital Networks ISDNs),*" Recommendation I. 120, Geneva, Switzerland (1993), and John Landwehr, "*The Golden Splice: Beginning a Global Digital Phone Network,*" Northwestern University (1992) each of which is incorporated herein by reference. ISDN replaces the existing analog local loop with a 160,000 bit/second digital connection. Since the bulk of long-distancc and inter-office traffic is already carried digitally, this digital local loop can be used

for end-to-end digital voice, computer data or any other type of information transfer. However, to achieve these data transmission rates on the local loop, special equipment must be installed at both ends of the line. Indeed, the entire telephone network is currently undergoing a transformation from a voice transmission network to a general data transmission service, with voice just being one particular form of data.

Once installed, each basic ISDN link will offer two data channels capable of 64,000 bits/second, a control channel with a capacity of 16,000 bits/second, reduced call connection time, and other benefits. At these rates, facsimile and still image transmission will be nearly instantaneous, high-quality audio will be feasible, and remote computer connections will benefit from a fivefold speed increase. Some progress toward full-motion video may also be achieved.

The down side of ISDN is its availability or lack thereof. To use ISDN, the user's central office must be upgraded to provide this service, the user must replace its on-premises equipment (such as telephones) with their digital equivalents, and each individual line interface at the central office must be modified to carry the digital data stream. This last step, the conversion to a digital link of the millions of analog connections between every telephone and the central office, is formidable. The magnitude of this task dictates that the deployment of ISDN will be slow and coverage will be sporadic for some time to come. Rural and sparsely populated areas may never enjoy these services.

Another existing infrastructure potentially capable of providing high-speed data communications services is the cable television system. Unlike the telephone system, which connects to users via low-bandwidth, twisted-pair wiring, the cable system provides high-bandwidth connectivity to a large fraction of residences. Unused capacities on this wiring could provide data rates of tens, or even hundreds, of millions of bits per second. This would be more than adequate for all of the services envisioned above including full-motion digital video. However, the cable system suffers from a severe problem --- its network architecture. The telephone system provides point-to-point connectivity. That is, each user has full use of the entire capacity of that user's connection---it is not shared with others and does not directly suffer due to usage by others. The cable system on the other hand, provides broadcast connections. The entire capacity is shared by all users since the same signals appear at each user's connection. ,Thus, although the total capacity is high, it is divided by the number of users requiring service. This architecture works well when all users require the same data, such as for cable's original design goal, television distribution, but it does not serve well a community of users with different data needs. In a metropolitan area the data capacity available to each user may be significantly less than via an ISDN or modem connection.

To provide high-speed, data connectivity to a large number of users, the cable system could be modified to isolate different segments of the user population effectively sharing the cable bandwidth over smaller populations. However, like ISDN, this will be a slow, costly process that will provide only partial service for many years to come.

The methods used to design modems are based largely on models of the telephone system that have remained unchanged for several decades. That is, a modem is modeled as an analog channel with a finite bandwidth (400-3400 Hz) and an additive noise component on the order of 30 dB below the signal level. However, a large portion of the telephone system now uses digital transfer of a sampled representation of the analog waveforms for inter-office communications. At each central office, the analog signal is converted to a 64,000 bit/second pulse code modulated (PCM) signal. The receiving office then reconstructs the analog signal before placing it on the subscriber's line. Although the noise introduced by this procedure is, to a first approximation, similar to that observed on an analog system, the source of the noise is quite different. See K. Pahlavan and J.L. Holsinger, *"A Model for the Effects of PCM Compandors on the Performance of High Speed Modems,"* Globecom '85, pages 758-762, (1985), which is incorporated herein by reference. Most of the observed noise on a telephone connection that uses digital switching is due to quantization by the analog-to-digital converters needed to convert the analog waveform into a digital representation.

As noted above, most telephone connections are currently carried digitally between central offices at rates of 64,000 bits/second. Furthermore, ISDN services demonstrate that it is possible to transmit significantly more than these rates over the local loop. It has been suggested that it may be possible to design a transmission scheme that takes advantage of these factors. Kalet et al. postulate a system, shown in Figure 2, in which the transmitting end selects precise analog levels and timing such that the analog-to-digital conversion that occurs in the transmitter's central office might be achieved with no quantization error. I. Kalet, J.E. Mazo, and B.R. Saltzberg, *"The Capacity of PCM Voiceband Channels," IEEE* International Conference on Communications '93, pages 507-51 1, Geneva, Switzerland (1993), which is incorporated herein by reference. By making use of the mathematical results of J.E. Mazo it is conjectured that it should be theoretically possible to reconstruct the digital samples using only the analog levels available at the receiver's end of the second local loop in the communications path. J.E. Mazo, *"Faster-Than-Nyquist Signaling,"* Bell System Technical Journal, 54:1451-1462 (1975), incorporated herein by reference. The resulting system might then be able to attain data rates of 56,000 to 64,000 bits/second. The shortcoming of this method is that it is nothing more than a theoretical possibility that may or may not be realizable. Kalet et al. state that *"This is a*

hard practical problem and we can only conjecture if a reasonable solution would be possible. " Id. at page 510.

An example of a conventional attempt to solve the foregoing problem is found in work by Ohta, described in US Patent Numbers 5,265,125 and 5,166,955, which are hereby incorporated by reference. Ohta disclosed an apparatus to reconstruct a PCM signal transmitted through a communications channel or reproduced from a recording medium. These patents "exemplify some conventional techniques abundant in the literature to deal with the general problem of reconstructing a multi-valued signal that has passed through a distorting channel." See also, for example, Richard D. Gitlin, Jeremiah F. Hayes and Stephen B. Weinstein, *"Data Communications Principles,* " Plenum (1992), incorporated herein by reference. However, such conventional teachings do not consider the application of methods to handle the output from a nonlinear quantizer, nor do they deal with the specific problems of decoding digital data passed over a telephone local loop. Furthermore, the problem of reconstructing a sampling rate clock from the PCM data is non-trivial when the PCM signal can take on more than two values. For example, in the patents by Ohta, a simple clock recovery scheme which relies on a binary input signal is employed. This type of clock recovery cannot be used with the multivalued codes used in a telephone system. Also, compensation for drift with time and changing line conditions requires use of an adaptive system which the prior art of PCM reconstruction does not include.

Thus, there is currently a critical disparity between the required or desired data communications capacity and that which is available. Existing modems do not provide adequate capacities, and new digital connectivity solutions are several years away from general availability. Refitting the existing infrastructure with ISDN capability is a sizable task and may take a decade before its use is widespread. A new method of data transmission could immensely benefit many current applications as well as making several new services available which would otherwise have to wait until the infrastructure catches up with the requirements.

Accordingly, there is a need for providing a new system of data transfer which provides the capability to receive data at high rates over existing telephone lines.

There is also a need for an improved system of data transfer which can enable systems, equipment, and applications designed for a digital telephone system (such as ISDN) to be used with analog connections.

There is also a need for an improved system of data transfer which is capable of taking advantage of the digital infrastructure of the telephone system without requiring costly replacement of all subscribers' lines.

It also would be desirable to create a high speed communication system to provide a means to distribute high-quality digital audio, music, video, or other material to consumers. Such an improved system of data transfer would advantageously provide a means to distribute, on-demand, individually-tailored information, data, or other digital material to a large number of consumers.

There is also a need for an improved high speed communications system to provide greater throughput for commercial applications such as facsimile, point-of-sale systems, remote inventory management, credit-card validation, wide-area computer networking, or the like.

SUMMARY OF THE INVENTION

One aspect of the present invention comprises a system for transferring data over existing telephone connections at rates higher than known modems or conventional methods of data transmission. The present invention achieves a significant improvement over conventional methods by making use of two critical observations:

1. The underlying phone system is digital using PCM transmission.
2. High data rates are required in one direction only, the source of which has direct digital access to the telephone system.

An aspect of the present invention utilizes the foregoing observations to achieve higher data transmission rates than were previously attainable with conventional systems. The second observation above addresses the largest use of modems --- to access and retrieve information from centralized servers. In addition, the present invention has been found particularly useful for applications that require higher data rates, such as database access and video or audio on demand. Such applications can be realized, utilizing the high data transmission rates that are attainable through the present invention.

An important aspect of the invention is both simple and extremely powerful; that is, to allow the data provider to connect directly to a digital telephone network while the consumer uses its existing analog connections without change to the line. This configuration greatly changes the model under which the consumer's data equipment must operate. Existing modems must deal with bandwidth limitations and multiple unidentified noise sources that corrupt a signal over the entire

transmission path. In contrast, an aspect of the present invention carries data digitally over the bulk of the path, from the central office to the consumer's home or office, and converts it to analog form only for the last segment of that path. Advantageously, one of the primary sources of noise to existing modems, quantization noise during analog-to-digital conversion, is completely eliminated since such conversion is no longer required. Furthermore, quantization noise during digital-to-analog conversion, can be modeled as a deterministic phenomenon, and thus significantly reduced.

By using the foregoing aspect of the invention, the data source, which has direct access to the digital network (for example, by ISDN), can transfer exact data to the central office serving the consumer of the data. All that is then required is a device at the consumer's end of the local loop that will compensate for distortion of the data signal due to the filtering performed at the central office's digital-to-analog converters and due to the transmission line. Both distortions can be dealt with adequately using existing digital signal processing hardware, as will be described herein.

Note that although this method cannot be used for return data from the consumer to the server, existing modems can be used, giving an asymmetrical channel with a capacity of up to 64,000 bits/second from server to consumer and 20,000 to 30,000 bits/second return.

It will be appreciated that an aspect of the invention enables digital data of any type (audio, video, information, or the like) to be sent to individual users at speeds higher than can be obtained with conventional modems, or conventional methods of data transfer. Furthermore, unlike cable television distribution systems, this invention can service, at the full data rate, any number of users simultaneously requesting different data.

Beyond providing greater speed of operation for existing applications, such as remote computer access, high-speed facsimile transmission, etc., certain aspects of the invention make several new applications possible. These include high-quality audio or music transmission, video-on-demand, still picture transmission, videophone, teleconferencing, or like applications in which high data transmission rates are essential.

Another aspect of the present invention is to reconstruct a multivalued PCM data signal from an analog representation of that signal. This is achieved using a new method which combines a novel clock synchronization technique with adaptive equalization.

In addition to the foregoing, other aspects and advantages of the present invention include: (1) the ability to effectively reconstruct the telephone system's digital pulse-code-modulated (PCM) data stream using only the analog signal at the subscriber end of the telephone line; (2) the ability to reconstruct the clock frequency and phase of the PCM data, using only the analog signal at the subscriber end of the telephone line; (3) the ability to increase the effective data rate between a central office and the subscriber end without adding additional equipment at the central office or otherwise modifying the telephone system; and (4) the ability to reconstruct said digital data after such data has been modified due to one or more of conversion to analog form, filtering, distortion, or corruption by the addition of noise.

BRIEF DESCRIPTION OF DRAWINGS

These and other features, aspects and advantages of the present invention will become better understood with regard to the following descriptions, appended claims and accompanying drawings in which:

Figure 1 is a block diagram showing a typical modem data connection;

Figure 2 is a block diagram showing an example of a hypothetical symmetric digital system,

Figure 3 is a block diagram showing a high speed distribution system in accordance with an aspect of the present invention;

Figure 4 is a block diagram of a hardware implementation of an encoder 150 of Figure 3, in accordance with an aspect of the present invention;

Figure 5 is a block diagram showing the function of encoder 150 of Figure 3, in accordance with an aspect of the present invention;

Figure 6 is a block diagram showing the function of a DC eliminator 184 of Figure 5, in accordance with an aspect of the present invention;

Figure 7a is a graph of a data stream 100 as a function of time, such as would be applied to encoder 150 in accordance with an aspect of the present invention;

Figure 7b is a graph of a typical output from encoder 150 as a function of time, such as would be applied to a digital network connection 132 of Figure 3, in accordance with an aspect of the present invention;

Figure 7c is a graph of a linear value 194 of Figure 6 as a function of time, this is the output signal from encoder 150 after conversion to linear form, in accordance with an aspect of the present invention;

Figure 8 is a block diagram showing the function of existing digital line interfaces, for reference in understanding an aspect of the present invention;

Figure 9 is a block diagram of a hardware implementation of a decoder 156 shown in Figure 3, in accordance with an aspect of the present invention;

Figure 10 is a block diagram showing the function of decoder 156 of Figure 3, in accordance with an aspect of the present invention;

Figure 11a is a graph of an analog signal 154 of Figure 10 as a function of time, in accordance with an aspect of the present invention;

Figure 11b is a graph of a compensated signal 2 74 of Figure 10 as a function of time, formed within decoder 156 in accordance with an aspect of the present invention;

Figure 11c is a graph of an estimated code stream 280 of Figure 10 as a function of time, formed within decoder 156 in accordance with an aspect of the present invention;

Figure 11d is a graph of a data stream 126 of Figure 3 as a function of time, generated by decoder 156 in accordance with an aspect of the present invention;

Figure 11e is a graph of an error signal 272 of Figure 10 as a function of time, generated by decoder 156 in accordance with an aspect of the present invention;

Figure 12 is a block diagram showing an inverse filter 268 of Figure 10, in accordance with an aspect of the present invention;

Figure 13 is a block diagram showing a feed-forward equalizer 300 of Figure 12, in accordance with an aspect of the present invention;

Figure 14 is a block diagram showing a filter tap 330 of Figure 13, in accordance with an aspect of the present invention;

Figure 15 is a block diagram showing a clock estimator 264 of Figure 10, in accordance with an aspect of the present invention;

Figure 16 is a block diagram showing the function of a clock synchronizer 260 of Figure 10, in accordance with an aspect of the present invention;

Figure 17 is a block diagram showing an end-to-end asymmetric system with a reverse channel in accordance with an aspect of the present invention;

Figure 18 is a block diagram showing an application of an aspect of the present invention with a database server;

Figure 19 is a block diagram showing an aspect of the present invention in an application to a high speed facsimile system;

Figure 20 is a block diagram showing a digital telephony relay in accordance with an aspect of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Conventional Modem Data Connection

A conventional modem data connection is shown in Figure 1. Operation of such a system is well known and has been standardized by government agencies such as the International Telecommunications Union. Depending on the types of a modem 104 and a modem 124, data may be applied at rates of up to 28,800 bits/second via the first user's data stream 100. Modem 104 converts data stream 100 into an analog signal, which is applied to a local loop 106, which in turn connects to a telephone switch 108. The analog signal is then carried through a telephone network 114 via a network connection 112 and eventually reaches, via a network connection 118, a telephone switch 120 serving the second user. The signal is then passed, in analog form, via a local loop 122 to the second user's modem 124, which converts the signal to data stream 126, which will be a delayed version of data stream 100. In an exactly analogous way, a data stream 128 travels through the telephone network via modem 124, local loop 122, telephone switch 120, a network connection 116, telephone network 114, a network connection 110, telephone switch 108, local loop 106, and modem 104 to form a delayed version as a data stream 102.

This system assumes that the telephone system reproduces the analog signal, applied at one user's telephone connection, at the other user's end with distortion and delay not greater than a set of standard values specified for the telephone system. One can show that, based only on these values, it is not possible to transmit data at rates greater than approximately 35,000 bits/second. This system ignores many details of the distortion, which may, in fact, be deterministic changes to the signal rather than unpredictable changes. One such deterministic change is quantization noise if telephone network 114 is implemented digitally. Existing modems cannot make use of knowledge of this significant noise source in eliminating distortion and are thus limited in their data rates. This is the key shortcoming of existing modem systems --- low data rate and a theoretical limit on the maximum improvement that will ever be possible within the current framework of assumptions.

In an attempt to overcome the foregoing shortcomings and disadvantages of a conventional modem data connection as shown in Figure 1, an approach to increasing the rate of data transfer has resulted in a hypothetical symmetric digital communication system, Such a system is shown in combination with a digital telephone network in Figure 2.

This system, described by Kalet et al. in the previously cited reference, is similar to existing modems but with a new assumption; that the underlying infrastructure is a digital telephone network 134. The operation is similar to that of the conventional modem system described above except that the signals are carried-in digital form within digital telephone network 134 and on a digital network connection 130, digital network connection 132, a digital network connection 136, and a digital network connection 138. Each user still requires a modem to transfer the information via local loop 122 and local loop 106 to telephone switch 120 and telephone switch 108 respectively where conversion between analog and a standard digital format used by digital telephone network 134 is performed.

Unlike conventional modems, no theoretical argument has yet been found which would limit the speed of such a system to less than that used internally within digital telephone network 134, typically 56,000 or 64,000 bits/second. Thus, it is hypothetically possible that such a system could obtain data rates up to 64,000 bits/second. However, such a system has never been reduced to practice nor is there any evidence that it would be possible to implement such a system. The authors of this system state that *"This is a hard practical problem and we can only conjecture if a reasonable solution would be possible."*

The problem is that to make use of the knowledge that the underlying network is digital and a large part of the observed signal distortion is due to quantization noise, the transmitting modem must control, via only its analog output, the digital levels chosen by the network to encode the signal. Furthermore, the receiving modem must, via only its analog input, accurately infer those digital levels. Distortion due to analog/digital conversion occurs at both the transmitter and receiver's end yet only the combined distortion added to the desired signal is directly observable. Furthermore, additional distortion due to electrical noise and crosstalk also occurs on local loop 122 and local loop 106. Separating out these distortion components from the desired signal and each other is a difficult, perhaps impossible, task.

One aspect of the present invention is a method by which the shortcomings of this approach are eliminated. It makes use of knowledge of the underlying digital network in a way that is realizable, providing higher attainable data rates than possible with any other known solution.

Sampling Rate Conversion

As will be seen in subsequent discussion, a system for recovering PCM data from a distorted analog representation requires a method of synchronizing the decoding clock with that used to convert the PCM data from a digital stream to analog values. Digital implementations of this synchronization require that a digital data sequence be resampled, changing its rate from that used by an analog-to-digital converter to one which is closer to that used in conversion from PCM data. Previously known techniques for achieving this are either strictly limited in their capabilities, or are computationally intensive. See, for example, R.E. Crochiere and L.R- Rabiner, *"Multitrate Digital Signal Processing,"* Prentice-Hall, Englewood Cliffs, NJ, 1983, which is hereby incorporated herein by reference. Performing sampling rate conversion between two independent clocks whose relationship may change as a function of time further complicates the task.

One aspect of the present invention is a method which can perform such conversion with a minimum of computational overhead. It accepts a continuously-variable input/output sampling rate ratio and performs the conversion with high accuracy. The techniques described can obtain greater than 9OdB anti-aliasing rejection and can be implemented in real-time on existing processors.

Overall System

Figure 3 shows an overview of the proposed system. The method of use of the system shown in Figure 3 is identical to that for current data communications circuits or modems. Data applied at data stream 100 will appear some time later at data stream 126. Data stream 100 to applied to encoder 150 whose function is to convert the data stream into a format compatible with the telephone system. The converted data is applied to digital telephone network 134 via digital network connection 132. The converted data emerges verbatim via digital network connection 138 at a client's telephone central office where a line interface 140 is located. At this point, if the client also had direct digital access to the digital connection to the chent's line interface from digital network connection 138, the transmission would be complete. However, where the client, like the majority of users, does not have direct digital access to the telephone network, this is not possible, and the following additional operations are required.

Line interface 140 converts the digital data on digital network connection 138 into an analog form in a manner conforming to the standardized specifications of digital telephony. The analog form is carried on local loop 122 to the client's premises where a hybrid network 152 terminates the line and produces analog signal 154. Hybrid network 152 is a standard part which converts the two-wire

bi-directional signal to a pair of one-way signals. Decoder 156 uses analog signal 154 to estimate and compensate for the distortion introduced by the conversion to analog form performed by line interface 140, resulting in an estimate of the digital data at digital network connection 138, which is assumed to be identical to the digital data that was applied at digital network connection 132. The transformation performed by encoder 150 is then inverted and decoder 156 outputs data stream 126, which is delayed estimate of the original data stream 100.

Note that within Figure 3, all elements are well known and exist within current digital telephone systems except encoder 150 and decoder 156, which will be described in detail below. Also to be described below, is a method of initializing and adapting decoder 156 to the exact conditions encountered in normal operation.

Physical Implementation Of Encoder

Figure 4 shows a block diagram of one possible realization of encoder 150 of Figure 3. Data stream 100 from Figure 3 is applied to the serial data input of a digital signal processor 160 such as an AT&T DSP32C. This processor uses a processor bus 162 to communicate with a read-only memory 168, a random access memory 166, and an ISDN interface circuit 164 such as an Advanced Micro Devices Am79C30A. Read-only memory 168 contains a stored-program whose functional characteristics will be described in following sections. Random acccss memory 166 is used for program storage and parameters. ISDN interface circuit 164 also has an ISDN connection 170, which is connected to a network terminator 172, such as Northern Telecom NTI, and subsequently to digital network connection 132, which was also shown in Figure 3.

To produce a fully-functional implementation, additional secondary elements such as decoders, oscillators, and glue logic would need to be added to the basic block diagram shown in Figure 4. Such additions are well known and will be evident to those skilled in the art.

Subsequent discussion of encoder 150 will refer to functional rather than physical components, all of which can, for example, be implemented as programs or subroutines for digital signal processor 160 using well-known digital signal-processing techniques.

Encoder Operation

Figure 5 shows a functional block diagram of encoder 150 of Figure 3. The channel from server to client begins with arbitrary digital data provided as data stream 100. Encoder 150 converts this bitstream into a sequence of eight-bit words sampled, preferably, at the telephone system's clock rate of 8,000 samples/second. This is achieved by a sequence of operations beginning with a serial-to-parallel converter 180, which groups together each eight bits read from data stream 100, outputting a stream of parallel eight-bit values as an 8-bit code stream 182. This mapping may preferably be performed such that the first of each eight bits read from data stream 100 is placed in the least-significant bit position of 8-bit code stream 182 with subsequent bits occupying consecutively more significant bit positions until the output word is complete, at which point the process repeats. DC eliminator 184 then inserts additional eight-bit values at regular intervals, preferably once per eight samples, such that the analog value associated with the inserted value is the negative of the sum of all prior values on 8-bit code stream 182. This is necessary since telephone systems frequently attenuate or remove any DC bias on a signal. DC eliminator 184 is one example of a circuit means for reducing DC components in the received analog signal.

A detail of the functional elements of DC eliminator 184 of Figure 5 is shown in Figure 6. A code stream 186 output from a two-input selector 190 is also converted to linear value 194 by a μ-law-to-linear converter 192, which can be implemented as a 256-element lookup table using the standard μ-law-to-linear conversion table. Values of linear value 194 are accumulated and negated by a summer 196 and a unit delay 200 to form a DC offset 198 and a previous DC offset 202, which is the corresponding unit-delayed value. DC offset 198 is applied to a linear-to-μ-law converter 204, which can use the same lookup table as μ-law-to-linear converter 192, but performing the inverse mapping. Note that if DC offset 198 is greater than or less than the maximum or minimum value in the table, the respectively largest or smallest entry will be used. A DC restoration code 206 is produced by linear-to-μ -law converter 204 and applied as one input to two-input selector 190. Two-input selector 190 operates by reading, preferably seven, sequential values from 8-bit code stream 182 and outputting these values as code stream 186, followed by reading and outputting a single value from DC restoration code 206. It then repeats this sequence of operations continually.

Returning to Figure 5, code stream 186 is applied to the input lead of an ISDN converter 188, which provides the well-known conversion to an ISDN signal. The function of ISDN converter 188 is implemented directly by several existing integrated circuits, including an Advanced Mcro Devices

Am79C30. The output of ISDN converter 188 forms digital network connection 132, which is also the output of encoder 150 of Figure 3.

For further understanding, some of the signals used by encoder 150 are illustrated in Figures 7a through 7c. Figure 7a shows a sequence of samples of data stream 100. After processing by serial-to-parallel converter 180 and DC eliminator 184, code stream 186 is shown in Figure 7b. Within DC eliminator 184, the linear equivalent of code stream 186, namely linear value 194, is shown in Figure 7c.

Line Interface

For reference during subsequent descriptions, Figure 8 shows a functional model of line interface 140 of Figure 3, such as would be found in a typical telephone system for use with an aspect of the present invention. Note that such interfaces are well known and are currently used in digital telephone switches. Digital telephone network 134 of Figure 3 passes an eight- bit-per-sample, μ-law-encoded, digital data-stream via digital network connection 138 to a μ-law-to-linear converter 210, shown in Figure 8. μ-law-to-linear converter 210 implements the well-known μ-law-to-linear conversion, converting each sample to a linear value 212. Linear value 212 is then converted to an analog signal 216 by a digital-to-analog converter 214 that is sampled using a telephone system clock 236 in a well known manner. Although not shown in Figure 3 for reasons of clarity, telephone system clock 236 is generated by digital telephone network 134. Analog signal 216 is then smoothed by a lowpass filter 218 to form a filtered signal 220. The main purpose of lowpass filter 218 is to provide a low-pass function with a cutoff frequency of approximately 3100 Hz. The International Telecommunications Union has standardized the specifications for digital-to-analog converter 214 and lowpass filter 218 in International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), *"Transmission Performance Characteristics of Pulse Code Modulation,"* Recommendation G. 712, Geneva, Switzerland, September 1992, which is hereby incorporated by reference.

Filtered signal 220 is multiplexed onto local loop 122 by a four-to-two-wire converter 222. Local loop 122 is bi-directional; incoming signals on local loop 122 are applied to four-to-two-wire converter 222 and are output as an unfiltered signal 234. Unfiltered signal 234 is applied to a bandpass filter 232, which has also been standardized by ITU-T in the above cited reference. The output from bandpass filter 232, a filtered signal 230, is converted to a linear value 226 by an analog-to-digital converter 228. Linear value 226 is then converted to digital network connection 136 by a

linear- to μ-law converter 224, which implements the standard linear-to-μ-law conversion. Note that in the system shown in Figure 3, digital network connection 136 is not used and has been omitted for clarity.

Physical Implementation of Decoder

Figure 9 shows a block diagram of one possible realization of decoder 156 of Figure 3. Analog signal 154 from Figure 3 is sampled by an analog-to-digital converter 240, which exists as an integrated circuit, such as a Crystal Semiconductor CS5016. This uses a clock signal 244, preferably at 16 kHz, generated by an oscillator 242, to form a digital input signal 246, which is connected to a bank of digital signal processors 248, such as AT&T DSP32C's, via one of their serial digital input leads. The processors are also connected to each other and to a random access memory 254 and a read-only memory 252 via a processor bus 250. Read-only memory 252 contains a stored-program whose functional characteristics will be described in following sections. Bank of digital signal processors 248 produces data stream 126, which is the final output of decoder 156 of Figure 3.

To produce a fully-functional implementation, additional secondary elements such as decoders, oscillators, and glue logic would need to be added to the basic block diagram shown in Figure 9. Such additions are well known and will be evident to those skilled in the art.

Subsequent discussion of decoder 156 will refer to functional rather than physical components, all of which can, for example, be implemented as programs or subroutines for the bank of digital signal processors 248 using well-known digital signal-processing techniques.

Decoder Operation

Figure 10 shows the functional structure of decoder 156 of Figure 3. Analog signal 154 from Figure 3 provides the input data to decoder 156. Analog signal 154 is fed to analog-to-digital converter 240 and converted to digital input signal 246, preferably sampled at 16,000 samples per second with 16 bits per sample precision. Analog-to-digital converter 240 exists as an integrated circuit, such as a Crystal Semiconductor CS5016. Digital input signal 246 is then processed by clock synchronizer 260 which interpolates and re-samples digital input signal 246 at intervals separated by a period estimate 262 to produce a synchronized signal 266. The operation of clock synchronizer 260 will be detailed in following sections. Synchronized signal 266 is filtered by inverse filter 268, which will be described below, to reconstruct compensated signal 274. The purpose of inverse filter 268 is

to invert the transformation performed by line interface 140 of Figure 3 of which the primary component is lowpass filter 218 of Figure 8. Returning to Figure 10, inverse filter 268 also outputs a delay error estimate 270 giving the timing error inherent in synchronized signal 266, which will be used by clock estimator 264, described below, to compute the period estimate 262 used by clock synchronizer 260. A decision means is then used to convert compensated signal 274 to a sequence of values from a discrete set. As an example, compensated signal 274 is converted to the nearest equivalent eight-bit μ-law word using a linear-to-μ-law converter 276 to give estimated code stream 280. As described earlier, linear-to-μ-law converter 276 may be implemented as a simple lookup table.

During normal operation, a switch 292 gates estimated code stream 280 back as a desired output signal 286, which is converted back to a linear signal by a μ-law-to-linear converter 278 to form a linear value 284, μ-law-to-linear converter 278 can be implemented as a simple lookup table as earlier described. During initialization, switch 292 will be set such that a predetermined training pattern 288 (not shown in Figure 3) is gated to desired output signal 286. This usage will be described below.

Linear value 284 provides an estimate of the desired value of compensated signal 274. It is used to adaptively update inverse filter 268 such that compensated signal 274 is as close as possible to linear value 284. This adaptation is one example of a training means for adjusting the parameters of decoder 156, which will be further explained in the discussion of inverse filter 268 below. A subtracter 282 computes error signal 272 using compensated signal 274 and linear value 284. Error signal 272 is fed back to an input lead of inverse filter 268 in a feedback loop. Estimated code stream 280 is also passed through a data extractor 290, which inverts the transformations performed by encoder 150 of Figure 3, to form the decoder's final output data stream 126.

For purposes of understanding only, examples of some of the signals present in Figure 10 are plotted in Figures 11a through 11e. Figure 11 a shows a typical input analog signal 154 to decoder 156, as a function of time. During processing of this signal, decoder 156 forms compensated signal 274, which is illustrated in Figure 11b. This signal is further processed to form estimated code stream 280, shown in Figure 11c. Finally, data extractor 290 of Figure 10 outputs data stream 126 shown in Figure 11d. Error signal 272, formed for internal use within decoder 156, is shown in Figure 11e.

As mentioned above, analog-to-digital converter 240, subtracter 282, linear-to-μ-law converter 276, switch 292, and μ-law-to-linear converter 278, all of Figure 10, are well known and

may be easily implemented by anyone skilled in the art. Following discussion will expand upon the implementation and operation of the remaining blocks; inverse filter 268, clock estimate or 264, clock synchronizer 260, and data extractor 290.

Inverse Filter

Figure 12 shows the internal details of inverse filter 268 of Figure 10. Inverse filter 268 is an example of an equalization means, which operates by performing linear filtering operations on an input signal (synchronized signal 266), to produce an output signal (compensated signal 274). Inverse filter 268 also receives error signal 272 that indicates the mismatch between compensated signal 274 and a desired value. It uses error signal 272 to update its filtering function such that error signal 272 in minimized. Such adaptive filter structures are well known; See for example Richard D. Gitlin, Jeremiah F. Hayes and Stephen B. Weinstein, "*Data Communications Principles,*" 'Plenun (1992), incorporated herein by reference. However, for purposes of clarification we will describe herein a preferred implementation of inverse filter 268. In addition, inverse filter 268 forms delay error estimate 270, which is used by clock estimator 264 of Figure 10.

Synchronized signal 266 is fed to feed-forward equalizer 300, which produces a partially-compensated signal 302 while using a correction signal 324 to perform adaptive updates. The operation of feed-forward equalizer 300 will be described below. Feed-forward equalizer 300 also outputs delay error estimate 270, which will be used by clock estimator 264 of Figure 10. Partially-compensated signal 302 is subsequently down-sampled by a factor of two by a down-sampler 304 to form a down-sampled signal 306. Down-sampler 304 operates by repeatedly reading two consecutive values from its input lead and placing the first of these on its output lead, discarding the second value. Down-sampled signal 306 is then applied to a subtracter 308 to form compensated signal 274. Compensated signal 274 is used by subsequent stages in Figure 10 and is also fed into a unit delay 310 to form a delayed signal 312. Delayed signal 312 is then applied to the input lead of a feed-back equalizer 314 to form a distortion estimate 3116. Feed-back equalizer 314 is similar to feed-forward equalizer 300 and will be further described below. Distortion estimate 316 provides the second input to subtracter 308. Error signal 272 of Figure 10 is scaled by a constant factor at a gain element 318 of Figure 12 to form a correction signal 320, which is applied as a second input signal to feed-back equalizer 314. Feed-back equalizer 314 uses correction signal 320 to perform adaptive updates.

Error signal 272 is also up-sampled by a factor of two by an upsampler 326, which inserts a zero between each sample of error signal 272. Up-sampler 326 produces an up-sampled error signal

328, which is subsequently scaled by a gain element 322 to provide correction signal 324. The use of correction signal 320 and correction signal 324 by feed-back equalizer 314 and feed-forward equalizer 300 respectively will be described below. The values of the parameters kf and kb of gain element 322 and gain element 318 respectively may, preferably, be in the range 10-2 to 10-15. Optimal values may easily be obtained by those skilled in the art without undue experimentation.

Feed-forward and Feed-back Equalizers

Figure 13 shows the internal structure of feed-forward equalizer 300 of Figure 12. Feed-forward equalizer 300 is composed of, preferably 8 -- 128, identical copies of filter tap 330 connected in a chain. Any convenient number of taps can be implemented. The first filter tap 330 accepts synchronized signal 266 of Figure 12 and the last filter tap 330 outputs partially-compensated signal 302 used in Figure 12. Each intermediate tap takes two input signals: a primary input 332 and a target input 336, to form two output signals: a primary output 334 and a target output 338. Each filter tap 330 also provides, as an output signal, a tap weight 340, which is used by a delay estimator 342 to compute delay error estimate 270. During operation, each filter tap 330 performs adaptive updates using, as an input, correction signal 324.

Figure 14 shows the details of the function of each filter tap 330 of Figure 13. Each tap has two inputs, primary input 332 and target input 336, and provides two outputs, primary output 334 and target output 338, using standard signal processing blocks as shown in Figure 14. Primary input 332 is delayed by one sample by a unit delay 350 to form primary output 334. Meanwhile, primary input 332 is also multiplied by tap weight 340 using a multiplier 352 to give a weighted input 354. Weighted input 354 is added to target input 336 by a summer 356 to give target output 338.

Adaptive update of tap weight 340 is performed by multiplying correction signal 324 by primary input 332 using a multiplier 366. A multiplier output value 364 provides a tap error estimate and is subtracted from a previous value 360 to form tap weight 340 using a subtracter 362. Previous value 360 is formed by a unit delay 358 using tap weight 340 as input. Each filter tap 330 also outputs tap weight 340.

Returning to Figure 13, each filter tap 330 is fed to delay estimator 342. Delay estimator 342 calculates delay error estimate 270 of the overall filter using the equation:

$$\Delta = \frac{\sum_{i=1}^{i=N} i \cdot w_i}{\sum_{i=1}^{i=N} w_i} - \frac{N}{2}$$

where w_i is an abbreviation for the $i-th$ tap weight 340. In this way, delay estimator 342 provides an estimation means for determining a degree of error in period estimate 262 of Figure 10.

The above description of feed-forward equalizer 300 of Figure 10 also applies to feed-back equalizer 314. The structure and operation of feed-back equalizer 314 are identical to that of feed-forward equalizer 300, with the exception that delay estimator 342 is not needed, so there is no equivalent to the delay error estimate 270 output. Also, feed-back equalizer 314 may use a different number of taps than feed-forward equalizer 300, preferably between one-quarter and one-half the number. The optimal number of taps to use for both feed-forward equaker 300 and feed-back equalizer 314 can be easily obtained by one skilled in the art without undue experimentation.

Clock Estimator

Figure 15 shows the functional components of clock estimator 264 of Figure 10. Clock estimator 264 is one example of a circuit means that uses delay error estimate 270 to update period estimate 262. The signal input to clock estimator 264, delay error estimate 270, is scaled by a factor of k_1, preferably in the range 10^{-1} to 10^{-8}, but dependent on the accuracy of the clock used for analog-to-digital converter 240, by a loop gain 370 to form phase error 374. Phase error 374 is then filtered with loop filter 376 to form period offset 378. Loop filter 376 is a low-pass filter whose design will be evident to those skilled in the design of phase-locked loops. Period offset 378 is added to nominal period 380 by summer 372 to create period estimate 262. Nominal period 380 is the a priori estimate of the ratio of half of the sampling rate of analog-to-digital converter 240 of Figure 10 to the frequency of telephone system clock 236 of Figure 8. Since telephone system clock 236 and the clock used by analog-to-digital converter 240 are not derived from a common source, the exact ratio will differ very slightly from 1.0 for the preferred choices of parameters. During operation, period estimate 262 will refine and track this ratio using estimates of the current error provided by inverse filter 268 of Figure 10.

Clock Synchronizer

A functional block diagram of clock synchronizer 260 of Figure 10 is shown in Figure 16. The function of clock synchronizer 260 is to interpolate and resample its input signal (digital input signal 246) at intervals separated by period estimate 262. For example, if period estimate 262 had a value of 2.0, every second sample read from digital input signal 246 would be output as synchronized signal 266. If period estimate 262 is not an integer, then clock synchronizer 260 will be required to appropriately interpolate between input samples to form the output samples.

Clock synchronizer 260 performs one cycle of operation for each output sample required. Each cycle begins with an accumulator 424 reading the value of period estimate 262 of Figure 10. Accumulator 424 forms a running sum of all inputs values read and outputs this sum as a real-valued sample index 426. This is scaled by a factor of N^{μ}, preferably in the range of 10-400, using a gain element 428 to form an upsampled sample index 430. The optimal value of N^{μ} can easily be obtained by one skilled in the art without undue experimentation. An integer/fraction splitter 432 decomposes upsampled sample index 430 into a sample index 422 and a fractional value 414. For example, if upsampled sample index 430 had a value of 10.7, integer/fraction splitter 432 would set sample index 422 to 10.0 and fractional value 414 to 0.7.

One of the input signals applied to a sample selector 398 is formed by a string of operations starting with digital input signal 246. An upsampler 390 reads a value from digital input signal 246 and outputs N^{μ} samples consisting of the value read from digital input signal 246 followed by $N^{\mu}-1$ zero values. The output stream from upsampler 390, an upsampled input signal 392, is applied to a low-pass filter 394, which has a passband cutoff frequency equivalent to 4 kHz. The design of upsampler 390 and low-pass filter 394 are well known. See, for example, R.E. Crochiere and L.R, Rabiner, *"Multirate Digital Signal Processing "* Prentice-Hall, Englewood Chffs, NJ, 1983, which is hereby incorporated herein by reference. Low-pass filter 394 forms a filtered upsampled signal 396, which is used as an input to sample selector 398.

Sample selector 398 is an example of a selection means, which reads a value from sample index 422 and interprets this as a sample number, s^n. It also maintains an internal count of how many samples it has read from its input lead connected to filtered upsampled

signal 396 since the system was initialized. It then reads additional samples from filtered upsampled signal 396 and forms output samples such that a sample 400 is a copy of sample s^n read from filtered upsampled signal 396 and a sample 402 is a copy of sample s^n+1. Sample 400 is then scaled by fractional value 414 using a multiplier 404 to form a sample component 408. Similarly, sample 402 is scaled by a fractional value 416 using a multiplier 406 to form a sample component 410. The magnitude of fractional value 416 is one minus the magnitude of fractional value 414, as computed using a subtracter 420, and a unit constant 418. Sample component 408 and sample component 410 are then added by a summer 412 to form synchronized signal 266, which is also the output of clock synchronizer 260 of Figure 10. The combination of multiplier 404, multiplier 406, and summer 412 is an example of an interpolation means for combining the samples selected by sample selector 398.

Clock synchronizer 260 can also be used in other applications or as a standalone sampling-rate converter. In general, synchronized signal 266 is equivalent to digital input signal 246 but with a different sampling rate. The ratio of the two rates is specified by period estimate 262 which may change as a function of time.

Note also that although the linear interpolation may appear to be a coarse approximation to the desired result, it is in fact quite accurate. By virtue of the oversampling performed by upsampler 390, filtered upsampled signal 396 has a frequency spectra that is near zero everywhere except for a narrow band around DC. The interpolation operation effectively creates images of this narrow passband in the frequency domain. The function of the linear interpolation is then to filter out these images. Conventional implementations use a sharp, computationally-expensive, low-pass filter to achieve this. Although the linear interpolator is a very poor low-pass filter, it does have very deep spectral notches at exactly the frequencies where the undesired images will appear. It is the combination of the placement of these notches with the narrow alias images that makes this method very accurate while eliminating much of the computation from traditional techniques.

Data Extractor

The last stage of decoder 156 of Figure 3 is data extractor 290 of Figure 10. The function of data extractor 290 is to invert the transformations performed by encoder 150 of

Figure 3. These transformations consist of serial-to-parallel converter 180 and DC eliminator 184 shown in Figure 5.

To invert these transformations, data extractor 290 first removes the values inserted into the data stream by DC eliminator 184. This is done by simply discarding every eighth sample read from the input (assuming the DC elimination was done by DC eliminator 184 using the preferred rate of once per eight samples). Once this is done, the stream of eight-bit values remaining can be converted back into a serial data stream 126 by outputting one bit of each word at a time, starting with the least-significant bit. Such techniques are well known by those skilled in the art.

Initialization of System

When a connection is first established between a server and a client, both encoder 150 and decoder 156 of Figure 3 must commence in a state known to each other. Within encoder 150 the following initiation is performed:

1. DC eliminator 184 of Figure 5 is initialized with two-input selector 190 of Figure 6 set such that its next output will be a copy of DC restoration code 206.

2. The output of unit delay 200 of Figure 6, previous DC offset 202, is initialized to 0.0.

3. Code stream 186 of Figure 5 is temporarily disconnected from DC eliminator 184. Instead a known sequence of N_c, preferably 16-128, values is repeated N_t, preferably 100-5000, times. The optimal values to use for N_c and N_t, can be easily obtained by one skilled in the art without undue experimentation.

The choice of N_c, above is tied to the design of decoder 156. N_c is preferably one-half of the number of taps in feed-forward equalizer 300 of Figure 12. Without loss of generality, one possible choice of the sequence of code values repeatedly transmitted by encoder 150 is shown in Table 1. An identical sequence is also used by encoder 150, applied as training pattern 288 in Figure 10.

Table 1: Typical Training Pattern

14	182	29	140	20	138	153	16
132	205	157	170	4	162	129	12
8	144	54	134	10	128	6	34
136	42	77	25	148	1	142	0

Once the N_t, repetitions of the sequence have been output, code stream 186 will be reconnected to DC eliminator 184 and subsequent output from decoder 156 will be reconnected to DC eliminator 184 and subsequent output from decoder 156 will correspond to the input applied as data stream 100 of Figure 3.

Within decoder 156 of Figure 3, the following initiation is performed before the first sample is read from analog signal 154:

1. Switch 292 of Figure 10 is set to gate training pattern 288 to desired output signal 286.

2. Data extractor 290 of Figure 10 is set so the next input value, estimated code stream 280, will be considered a DC equalization value and thus be discarded.

3. Unit delay 310 of Figure 12 is initialized to output zero as delayed signal 312.

4. Upsampler 326 of Figure 12 is initialized such that its next output, up-sampled error signal 328, will be a copy of error signal 272.

5. Downsampler 304 of Figure 12 is initialized such that its next input value, partially-compensated signal 302, will be copied out as downsampled signal 306.

6. Within feed-back equalizer 314 and feed-forward equalizer 300 of Figure 12, each unit delay 350 of Figure 14 is initialized to have a zero output.

7. Within feed-back equalizer 314 of Figure 12, each unit delay 358 of Figure 14 is initialized to zero.

8. Within feed-forward equalizer 300, each unit delay 358 of Figure 14 is initialized to zero.

9. Accumulator 424 of Figure 16 is initialized to output a value of zero as real-valued sample index 426.

10. Low-pass filter 394 is initialized with an all-zero internal state.

11. Upsampler 390 is initialized such that its next output, upsampled input signal 392, will be the value of digital input signal 246.

Decoder 156 then operates as described earlier until N_c - N_t values have been formed at estimated code stream 280 of Figure 10. At this point, switch 292 is moved to gate estimated code stream 280 to desired output signal 286. From this point on, data stream 126 should correspond to data read from data stream 128 as shown in Figure 3.

It must also be ensured that encoder 150 and decoder 156 enter and leave initialization mode such that the values on data stream 100 and data stream 126 of Figure 3 are in exact correspondence. One example of a method to achieve this synchronization is to violate the DC restoration performed by DC eliminator 184. To signal the beginning of training, code stream 186 is set to the maximum legal code value for longer than the normal DC restoration period, for example for 16 samples. This is followed by setting code stream 186 to the minimum legal code value for the same number of samples. The training pattern then follows this synchronization pattern. Similarly, the end of training can be signaled by reversing the order of the above synchronization pattern --- repeating the minimum value followed by the maximum value. These synchronization patterns can then be detected by decoder 156 and used to control switch 292.

Other techniques for such synchronization are well known and are used in existing modems. See, for example, ITU-T, V.34, previously cited.

Alternate Delay Estimator

In previous discussion, delay estimator 342 was formed by examination of the filter tap weights within feed-forward equalizer 300. Other delay estimation means are also possible. For example, error signal 272 and compensated signal 274 of Figure 10 can be used to form delay error estimate 270 as follows:

$$\Delta = \frac{e}{\frac{dv}{dt}+k}$$

where Δ is delay error estimate 270, v is compensated signal 274, e is error signal 272, and k is a parameter which can be easily obtained by those skilled in the art without undue experimentation. The value of k will depend upon the relative contributions of signal noise and clock jitter observed. Any other methods of implementing a delay estimation means to form delay error estimate 270 may also be used in the present invention.

Alternate Decoder Initialization Method

As described above, the parameters of decoder 156 may be established using fixed initialization values followed by a training period during which a known data sequence is transmitted. The previously described method uses the training sequence to perform sequential updates of the parameters of inverse filter 268 and clock estimator 264 on a sample-by-sample basis.

It is also possible to perform a single block update of all parameters. During the transmission of the training sequence, decoder 156 merely stores the values that appear as digital input signal 246. Once the entire training sequence has been transmitted, decoder 156 can perform an analysis of the acquired values and calculate values for its internal parameters.

The calculations needed to perform the parameter estimation are as follows:

1. Calculate the fundamental digital period, T_u, of the acquired signal using a rate estimation means. This can be done using any of a variety of well-known signal processing techniques, such as an autocorrelation analysis. It is known in advance that T_u is approximately twice N_c, the length of the training sequence, assuming the use of the preferred sampling rate for analog-to-digital converter 240. The only

source of difference will be due to differences between the sampling rate of telephone system clock 236 and half the sampling rate of analog-to-digital converter 240.

2. Initialize nominal period 380 of Figure 15 as $\dfrac{T_u}{2 \cdot N_c}$.

3. Resample digital input signal 246 by passing it through clock synchronizer 260 with delay error estimate 270 set to zero, to form synchronized signal 266.

4. Form a matrix Y with 2 - N_c columns and N_t rows. The elements of Y are the values of synchronized signal 266 as computed above. These are stored in the matrix by filling the first row with sequential samples of synchronized signal 266, then the second row, and so on.

5. Compute the mean of each column of U to form r, a 2 • N_c element vector.

6. Compute an estimate of energy, σ^2, of the noise component of the input signal using:

$$\sigma^2 = \frac{1}{N_c \cdot N_t} \sum_{j=1}^{N_c} \sum_{i=1}^{N_t} (Y_{ij} - r_i)^2$$

where Y_{ij} is the element in column I, row j of Y.

7. Compute the N_c, element vector, c, by passing the training sequence values, such as those shown in Table 1, through a converter such as μ-law-to-linear converter 278.

8. Form a matrix, A, with $N_f + N_b$ columns and N_C rows as follows:

$$A = \begin{bmatrix} r_1 & r_2 & \cdots & r_{N_f} & c_{N_c-N_b+1} & c_{N_c-N_b+2} & \cdots & c_{N_c} \\ r_3 & r_4 & \cdots & r_{N_f+2} & c_{N_c-N_b+2} & c_{N_c-N_b+3} & \cdots & c_1 \\ r_5 & r_6 & \cdots & r_{N_f+4} & c_{N_c-N_b+3} & c_{N_c-N_b+4} & \cdots & c_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{2N_c-1} & r_{2N_c} & \cdots & r_{N_f+2N_c-2} & c_{N_c-N_b} & c_{N_c-N_b+1} & \cdots & c_{N_c-1} \end{bmatrix}$$

where N_f is the number of filter taps in feed-forward equalizer 300 of Figure 12 and N_b is the number of filter taps in feed-back equalizer 314. For example, if $N_c = 3$, $N_f = 4$, and $N_b = 2$, then:

$$A_{example} = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 & c_2 & c_3 \\ r_3 & r_4 & r_5 & r_6 & c_3 & c_1 \\ r_5 & r_6 & r_1 & r_2 & c_1 & c_2 \end{bmatrix}$$

9. Find the value of a $N_f + N_b$ element vector, x, which minimizes e^2 in the following equation:

$$e^2 = \frac{(Ax-c)^T(Ax-c)}{N_c} + \sigma^2 \sum_{i=1}^{N_f} x_i^2 + e^2 \sum_{i=N_f+1}^{N_f+N_b} x_i^2$$

This can be solved using well-known techniques from linear algebra, calculus and iterative methods, which will be obvious to those skilled in the art.

10. Initialize previous value 360 of Figure 14 for each tap of feed-forward equalizer 300 with $x_1 \ldots xN_f$ respectively.

11. Initialize previous value 360 for each tap of feed-back equalizer 314 with $xN_f + 1 \ldots xN_f + N_b$ respectively.

12. Once these parameters have been computed, normal operation can commence. Note that the parameters will subsequently change due to adaptive updates based on error signal 272, as previously discussed.

The above sequence should be viewed as an example of another method of doing initiation of decoder 156 using a training sequence. Other methods and numerous variants are also possible. For example, the received training sequence may be truncated at each end to remove effects of the transient in switching between normal and training modes; the exact transition levels in linear-to-μ-law converter 276 and μ-law-to-linear converter 278 may be adjusted using the training information, modified equations for each previous value 360 may be used, etc.

Preferred Alternate Training Procedure

The following is a description of the preferred steps in training the encoder 150 and the decoder 156, shown in Figure 3:

1. The encoder 150 sends a repeating pattern to the digital telephone network. This pattern consist of M repetitions of a sequence of N PCM codewords to givc a total of $M \times N$ codewords. The term "PCM codewords" as used herein refers the set of codewords that is utilized by the digital telephone network 134. The N PCM codewords are each chosen randomly from two values which are the negative of each other. For example, the PCM codewords 0x14 and 0x94 correspond to the negative of each other under the telephone network's μ-law companding rule. The random selection is also constrained such that each of the two PCM codewords is used exactly $N/2$ times. This guarantees that the training sequence will not have any DC component.

2. The decoder 156 receives the analog signal 154, which is the analog equivalent of the PCM codeword pattern, and stores it using prior knowledge of M and N. The analog signal 154 is sampled and stored at a nominal rate of 16000 samples/second.

3. The stored sequence is analyzed to find its repetition rate using standard signal processing techniques. Although the period associated with the repetition rate should be $N/8000$ seconds, there may be a slight discrepancy due to the difference in the exact values of the decoder's clock and the clock used by the telephone network 134. This discrepancy in repetition rate may then be used to adjust the decoder's clock and resample the stored signal using the corrected clock. This process can be repeated multiple times until the measured period exactly matches that expected.

4. The noise level is preferably measured by looking at the variance among the M repetitions of the test pattern. Each repetition should be identical and it is only the noise that causes

these variations. Some of the repetitions, including the first or last ones, may be ignored if they are significantly different from the mean. This may eliminate or decrease end effects and intermittent noise bursts or crosstalk.

5. The average signal level of the considered repetitions is then determined giving an average received sequence with length $2N$. The length $2N$ arises from sampling the incoming analog signal 154 at 16000 samples/second.

6. An optimal equalizer is designed using the known transmitted sequence, the average received signal level and the noise estimate. The equalizer with the minimum mean-squared error is found by solving a set of linear equations using well-known methods. In particular, a combination of a fractionally-spaced equalizer with 90 taps operating at 16,000 samples/second followed by a 20-tap decision-feedback equalizer operating at 8000 samples/second may be used.

7. The mean-squared error estimate and the equalizer settings may be transmitted back to the encoder 150 when a reverse channel is utilized, as described below. The encoder 150 then uses these values to choose a codeword set and encoding method that maximizes the throughput of the communication link.

After training is completed, the encoder 150 begins sending data via the link. The decoder 156 uses the computed clock adjustment and equalizer setting to compensate the received signal and then makes decisions as to which PCM codeword was sent by the encoder 150. The choice is then further processed to convert the PCM codeword into a data sequence. In addition, the measured deviation between the compensated signal and the nearest actual PCM codeword is used as a continuous error measure. This error measure can be fed back to the equalizer and clock adjustment circuitry in the decoder 156 to allow continuous updates and prevent any drift. The error measures may also be used to determine if the quality of the line changes significantly in which case the encoder 150 is notified that a retrain should be initiated.

Addition of a Reverse Channel Description

Figure 17 shows an aspect of the present invention that combines the previously described communication system with a reverse channel. Data stream 100 is applied to encoder 150 as was described in reference to Figure 3. This in turn connects to digital telephone network 134 via digital network connection 132. The data emerges verbatim from

thc network at the client's central office via digital network connection 138. The digital information is converted to analog form by line interface 140 and placed in analog form on local loop 122. At the client's premises, hybrid network 152 forms incoming analog signal 448 and an echo canceler 442 removes contributions to incoming analog signal 448 from an outgoing analog signal 444 to form analog signal 154. Analog signal 154 is then applied to decoder 156, which provides data stream 126. Data stream 128 from the client is converted to outgoing analog signal 444 by a modulator 446 in accordance with well-known techniques such as used in existing modems, and then applied to echo canceler 442 as well as fed onto local loop 122 via hybrid network 152. At the central office, this is converted to digital network connection 136 by line interface 140. Digital telephone network 134 transfers the data on digital network connection 136 to digital network connection 130. A demodulator 440 then converts this to data stream 102 for the server.

For systems using a reverse channel, such as is shown in Figure 17, the decoder 156 may be coupled to a conventional modulator 446, such as a V.34 modulator, to provide bidirectional communication. In this case, an echo canceler 442 is preferably used to prevent the output of the modulator 446 from appearing as an input to the decoder 156.

Although a conventional echo canceler may be used, the nature of the signals in the system described herein have special properties that may be advantageously used. Specifically, the incoming analog signal 154 may have frequency components from near DC up to 4 kHz, whereas the outgoing signal from the modulator 446 is more tightly bandlimitted to the range of 400 Hz to 3400 Hz. This asymmetry in bandwidths between the incoming and outgoing channels may be exploited by using an asymmetric echo canceler. Moreover, if the bandwidth of the outgoing channel is further reduced, the asymmetry will be even greater and the benefit of the asymmetric echo canceler will increase.

At the encoder 150 end of the connection, a digital echo canceler is preferably used between the encoder 150 and the demodulator 440 shown in Figure 17. Here again, the asymmetric form of the connection may be exploited by using an asymmetric echo canceler.

Operation

The system shown in Figure 17 provides full duplex communication between two telephone subscribers: one with digital connectivity, and the other with analog connectivity. The operation of the forward channel is as described above in reference to Figure 3, with one addition. Echo canceler 442, inserted between hybrid network 152 and decoder 156 has been

added to reduce the effects of the reverse channel. Echo canceler 442 scales outgoing analog signal 444 and subtracts it from a incoming analog signal 448 to produce analog signal 154. The techniques and implementation of echo cancelers are well known. The reverse channel can be implemented using a variant of existing modem technology. See, for example, International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), *"A Duplex Modem Operating at Signaling Rates of up to 14,400 Bit/s for Use on the General Switched Telephone Network and on Leased Point to Point 2-wire Telephone-Type Circuits* " Recommendation V.32bis, Geneva, Switzerland (1991), incorporated herein by reference. Data are modulated by modulator 446 to form outgoing analog signal 444 that can be carried by the telephone system. The modulation techniques that may be employed are well known. For example, methods capable of transfers at up to 14,400 bits/second are described above. Similarly, methods capable of transfer rates up to 28,800 bits/second are described in International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), Recommendation V.34 Geneva, Switzerland (1994), also incorporated herein by reference.

Outgoing analog signal 444 is placed on local loop 122, using hybrid network 152, such as is employed in virtually all telephone equipment. Hybrid network 152 converts between a four-wire interface (two independent, unidirectional signals) on one side and a two-wire interface (one bidirectional signal) on the other side. The two-wire signal is simply the sum of the two signals on the four-wire side. At the client's central office, the telephone company's equipment converts the analog signal on local loop 122 to digital network connection 136, which is sampled at 8,000 samples/second using telephone system clock 236. In North America, this conversion is performed to provide eight bits per sample using a nonlinear mapping known as μ-law to improve the signal-to-noise ratio of typical audio signals. Once converted to μ-law, the client's signal is carried by digital telephone network 134 until it reaches the server's premises. Note that since the server has a digital connection to the phone system, the signal is not converted to analog form by the server's central office. There may, however, be several layers of interfaces (such as ISDN 'U' or 'S', etc.) intervening between the server and digital network connection 136. However, since the same data presented at digital network connection 136 also appears at digital network connection 130 later, this intervening hardware can be ignored. Demodulator 440 performs the inverse function of modulator 446, as done by existing modems, with one small exception. Since both its input and output are digital, it can be implemented completely in digital hardware, whereas existing modems must work with an analog input. As with modulator 446, the

implementation of demodulator 440 is well known and is described in the literature such as International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), *"A Duplex Modem Operating at Signaling Rates of up to 14,400 bit/s for Use on the General Switched Telephone Network and on Leased Point-to-Point 2-wire Telephone-type Circuits,* " Recommendation V.32 bis, Geneva, Switzerland (1991). Note that even the reverse channel can exhibit performance superior to traditional modems since degradation of the signal will occur only at the consumer's local loop. Existing modems must deal with distortions occurring on local loops at both ends of the communications path. Alternative implementations of this invention may use other well-known methods or techniques to provide a reverse channel or may eliminate it altogether. Thus, the description of one possible reverse channel implementation is provided merely for illustration and should not be construed as limiting the scope of this aspect of the invention. Note that the provision of a reverse channel also simplifies the synchronization of decoder 156 and encoder 150 and allows the system to be re-initialized if needed. The performance of the system may be monitored by decoder 156 by examination of error signal 272 of Figure 10. If error signal 272 exceeds a given level, preferably one-third of the average difference between μ-law linear values, decoder 156 can notify encoder 150 via the reverse channel that the system should be re-initialized.

Combination with a Source Coder

It is possible to extend the function of encoder 150 and decoder 156 shown in Figure 3 to perform additional invertible transformations on data stream 100 before application to encoder 150. The effects of these transformations can be removed by applying the inverse transformation to the output of decoder 156 before producing data stream 126. This transformation advantageously may provide any invertible, function including, but not limited to:

Error Correction

Bits may be added to the data stream to provide error correction and/or detection using any of the well-known methods for such operations. These include, for example, convolutional coding, block codes or other error correction or detection schemes well documented in the literature. Note that if the same error processing applied to data stream 126 is also inserted in the signal path from linear-to-μ-law converter 276 to μ-law -to-linear converter 278, shown in Figure 10, the quality of desired output signal 286, linear value 284, and error signal 272 will be improved and the performance of decoder 156 will benefit.

Subset of Source Alphabet:

Although there are 256 possible μ-law codewords available for data transmission, the μ-law mapping results in these words being unequally spaced in the linear domain. Thus, some pairs of codewords will be more easily confused by decoder 156 due to line noise or other impairments. The source coder can restrict its output to a subset of these codewords to improve the accuracy of decoder 156 at the expense of reduced gross data rate. This can also be used to adapt decoder 156 to poor line conditions by reducing the codeword alphabet if the decoder detects that it is unable to separate codewords within a given error criterion. By reducing the codeword set, improved error margins will result at the cost of decreased data rate. Thus, the system can handle degraded connections by lowering the data rate.

In the systems shown in Figures 3, 17 and 18, data is transmitted as a sequence of the 8-bit PCM codewords used by the digital telephone network 134. In the simplest implementation, all 256 possible PCM codewords can be used and the data can be simply taken 8 bits at a time and used verbatim as the PCM words.

This scheme, however, has potential problems. First, in the standard μ-law interpretation of the PCM values, there are two codewords from the set of 256 that map onto the same analog value. Thus, it would be impossible to distinguish between these codewords using only the resulting analog signal, such as the analog signal 154. Second, the μ-law interpretation of the PCM codewords correspond to unequally spaced analog levels. Levels from the companding rule that are closely spaced are easily confused and should be avoided. Third, the digital telephone network 134 sometimes uses the least-significant bit of the PCM codewords for internal signaling making these bits unreliable. Fourth, when the PCM codewords are converted to their equivalent analog voltage level, they are passed through various filters, such as the CODEC's smoothing filter and the subscriber loop. The result of this is that some frequency components, notably DC and high-frequencies, are attenuated. If the decoder 156 is to equalize these frequency components, noise levels will be necessarily increased resulting in greater confusion between levels. Fifth, the digital telephone network 134 may perform a remapping of the codewords onto new values from the same set such as would be the case on an international call that uses both μ-law and A-law encoding, or if the network attempts to modify the signal levels by remapping.

The system shown herein may deal with these issues in two ways. First, the encoder 150 may use only a subset of the 256 PCM codewords. In addition, the encoder 150 may choose the PCM

codewords with reference to previously transmitted codewords in order to reduce certain frequency components in the resulting analog signal 154.

The first step in this process is to identify the frequency and noise characteristics of the transmission channel. The encoder 150 sends a known training pattern consisting of randomly chosen independent PCM codewords. When converted to an analog signal, such as the analog signal 154, this will result in an analog signal with an approximately flat frequency spectrum. The decoder 156 receives a distorted version of the analog signal 154 and constructs a synchronizer and equalizer that reduces this distortion. In the process, the decoder 156 obtains measurements of the noise levels and filtering characteristics of the line. These measurements are transmitted back to the encoder 150. These steps are described above under the heading "Preferred Alternate Training Procedure."

With the noise levels known, the encoder 150 may then choose a subset of the PCM codewords such that the probability of any one codeword being confused with another due to the noise is below some preset threshold probability. The subset is chosen by first calculating the required minimum separation between the codewords using the noise statistics. The codeword set is then chosen by selecting the largest possible set for which no pair of codewords is separated by less than the minimum preset threshold. The choice of codeword set may also be constrained to not use any codewords that will be corrupted by bitrobbing, as is further described below.

To eliminate certain frequencies from the analog signals reconstructed from the PCM codewords, one technique is to insert additional codewords at regular intervals that do not contain data. These insertions are used to shape the output spectrum. For example, if we wish to eliminate DC, then once for each N data-carrying codewords, a codeword is inserted whose analog value is as close as possible to the negative of the sum of the values of all previous codewords, as described above with reference to the DC Eliminator 184.

In general, the inserted codewords may be chosen to shape the spectrum as needed. For example, the codewords may be passed through a digital filter at the encoder 150, with the inserted codewords being selected to minimize the output energy of the filter. If the digital filter is chosen to pass components that it is desirable to eliminate, such as low frequencies or those near 4 kHz, then this procedure may minimize those components.

Since the data-carrying codewords are chosen from a subset of the 256 possible codewords, the data usually cannot simply be placed in the codewords 8 bits at a time. Instead, a group of bits from

the data are used to choose a sequence of codewords. For example, if the subset consists of only 3 codewords, then we could choose the value of groups of 4 codewords (of which there are $3^4 = 81$ possibilities) using groups of 6 input bits (of which there are 2^6 or 64 possibilities). In this case, there is some waste of the possible information content, but this decreases as the length of the groups are increased.

Use With 56,000 bit/second Telephone Systems

In some PCM transmission schemes used by the telephone systems, the least significant bit of each eight-bit codeword is used for internal synchronization. This can be handled by transforming data stream I 00 by inserting a zero bit once per eight bits such that the encoding process described in reference to Figure 5 will place the inserted bit into the least-significant bit position of each encoded value applied to digital network connection 132. These inserted zeroes will then be removed at decoder 156 by post-processing data stream 126. In this way, the telephone system's use of the low order bit will not damage the transmitted data, but the maximum data rate will be reduced to 56,000 bits/second.

On the long distance telephone transmissions, including transmissions over the digital telephone network 134, some traffic is carried on lines which use in-band signaling. In these cases, the digital telephone network 134 may use (or usurp) the least significant bit ("LSB") of every sixth PCM codeword for ring indication and other signals. This technique is commonly known as "Robbed-bit signaling" or simply "bitrobbing." If the telephone connection is used to carry data, then this implies that only 7 bits can be used during the bitrobbed frames. Since the sender has no control over the bitrobbing, one approach to deal with this problem is to never use the LSB, thereby reducing the maximum data rate to 56 kbps (7 bits/codeword x 8000 codewords/second).

The system shown in Figure 3, for example, may improve upon this by having the decoder 156 identify the bitrobbed frames and then direct the encoder 150 to avoid the LSB only in the frames where bitrobbing is occurring. Thus, a single hop which uses bitrobbing will reduce the bit rate less; i.e to 62.7 kbps instead of 56 kbps. In the case of multiple hops over unsychronized links, bitrobbing may occur at a different phase resulting in further reduction in bit rate.

The detection of bitrobbed frames can be done during the initial training of the system. The encoder 150 sends a known pattern of 8-bit PCM codewords over the transmission channel and the decoder 156 stores samples from the resulting analog waveform it receives. The decoder 156 then attempts to resynchronize and equalize this signal to minimize the difference between its output and

the known pattern under the assumption that no bitrobbing occurred. The decoder 156 then measures the average equalized values at each of 6 phases. That is, the errors at the 1st, 7th, 13th, 19th, etc. samples are average as are the errors at the 2nd, 8th, 14th, etc. samples giving 6 average error measurements. The decoder 156 then makes a decision for each phase as to whether it has 1) not been bitrobbed, 2) been bitrobbed, with the LSB replaced by 0, 3) been bitrobbed, with the LSB replaced by 1, or 4) been bitrobbed, with the LSB replaced by ½. The choice is made by determining which bitrobbing scheme (1, 2, 3 or 4) minimizes the difference between the equalized signal and the known pattern after undergoing the given bitrobbing.

Once the bitrobbing that occurred in each frame is determined, the equalization process is rerun, since the first equalization was done without knowledge of the bitrobbing. This second pass will provide a better equalized signal. The bitrobbing decision can then be verified with the second equalized signal in the same manner as above.

Once the bitrobbing is known for each of the 6 phases, the decoder 156 preferably transmits this information to the encoder 150. In subsequent data transmissions the encoder 150 and decoder 156 may avoid using bits that will be corrupted by bitrobbing.

In an alternative approach, the encoder 150 sends a known pattern of codewords after the above described clock synchronization and equalization steps have been completed. The decoder 156 may then compute statistics regarding the received signal levels, such as the levels represented by the compensated signal 274 shown in Figure 10, for each of the transmitted 256 PCM codewords for each of the 6 phases. The computed statistics, including the mean signal level and the variance, may then be used to select a subset of codewords that provides a predetermined probability of error. The selected subset may then be transmitted to the encoder 150 and used in subsequent transmissions. In addition, the μ-law-to-linear and linear-to-μ-law converters 192, 276 and 278 may then use the mean levels computed above instead of the predetermined levels of the converters 192, 276 and 278. This method, therefore, provides adaptive converters that may handle implicitly, and without prior knowledge or explicit analysis, bitrobbing or other remapping that occurs in the digital telephone network 134.

Data Compression

The source coder may provide lossless (or lossy) compression of the data stream 100 using any of the various known techniques well known to those skilled in the art. These

include, but are not limited to, Lempel-Ziv compression, run-length coding, and Huffman encoding. The inversion of the chosen compression transformation, which is also well known, can be applied to data stream 126.

Use with Other Telephone Systems

The above methods can also be used with telephone systems that use nonlinear companding operations other than μ-law to transport the audio signal. For example, many parts of the world use a similar encoding, known as A-law. Aspects of the present inventions can be adapted to such systems by replacing all μ-law -to-linear and linear-to-μ-law converters with their A-law equivalents. These equivalents can also be implemented using a 256-element lookup table. In this case the table would be populated with the well-known A-law mapping. These modifications will be evident to those skilled in the art.

Combination with Existing Modems

An aspect of the present invention may also be used in conjunction with existing modems. In a traditional system, such as shown in Figure 1, modem 104 may be modified to also incorporate the functionality of encoder 150 described above. Furthermore, modem 124 may be modified to also include the functionality of decoder 156. When a call is connected between the modified modem 104 and modem 124, both operate as for a normal connection between unmodified modems. After they have completed their initiation, modem 104 can send a negotiation request to modem 124 using well-known negotiation protocols such as those standardized by the International Telecommunications Union. If modem 124 includes an implementation of decoder 156, it can respond positively to the request. Otherwise the request will be rejected and normal modem communications will be used. Once a positive response has been received, modem 124 and modem 104 can switch to operating as shown in Figure 17, beginning with an initialization sequence. In this way, a combined modem/decoder can interoperate with existing modems and, when possible, also advantageously provide increased throughput using an aspect of the present invention.

Combination with a Database Server

An aspect of the present invention may be used with a central server to provide data communications of any type (information, audio, video, etc.) between a central site and multiple users as shown in Figure 18. A server 450 provides a server data 452 to a server interface 454 which consists of an array of encoders, such as encoder 150 described herein, and, possibly, an array of demodulators such as demodulator 440. Server interface 454

connects to digital telephone network 134 via a server connection 456 such as an ISDN PRI interface. Each subscriber to the service has a client interface 460 consisting of decoder 156 and, optionally, echo canceler 442 and modulator 446 similar to those shown in Figure 17. Client interface 460 operates on a client connection 458 to provide a client data stream 462. Overall, this configuration allows multiple users to independently communicate with a central server or servers. This configuration is usable for any type of data service including, but not limited to: audio or music distribution, on-line services, access to networking services, video or television distribution, voice, information distribution, credit-card validation, banking, interactive computer access, remote inventory management, point-of-sale terminals, multimedia. Other implementations or configurations of this invention are also applicable to these and other applications.

High-Speed Facsimile Transmission

An aspect of the present invention. shown in Figure 19, may be used for high-speed transmission of facsimiles. A transmitting FAX 470 scans an image and translates it into a transmitted data stream 472 in a well-known manner. Transmitted data stream 472 is transmitted to a received data stream 476 via a distribution system 474 as shown, for example, in Figure 17. A receiving fax 478 converts the data stream back into an image and prints or otherwise displays it. Distribution system 474 may be implemented as shown in Figure 17 with data stream 100 replaced by transmitted data stream 472 and data stream 126 replaced by received data stream 476. Furthermore, data stream 128 and data stream 126 may be used for protocol negotiations between receiving fax 478 and transmitting FAX 470 as described in International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), Recommendation V.17, "A 2-Wire Modem for Facsimile Applications With Rates up to 14,400 b/s, " Geneva, Switzerland (I991) which is hereby incorporated herein by reference. In this way, facsimiles from transmitting FAX 470 can be advantageously transmitted to receiving fax 478 at rates higher than possible using conventional transmission schemes.

ISDN/Digital Telephony Relay

An aspect of the present invention can also be used in conjunction with any application that can make use of ISDN or digital telephony. This can provide a functional equivalent to ISDN for transmission from a digitally connected party to a second party who has only analog connectivity to the telephone network. This could be done either directly using a system such as shown in Figure 17, or by use of a mediating relay as shown in Figure

20. A digital subscriber 480 can make a digital call to an analog subscriber 490, who does not have direct digital access to the digital telephone network but has instead an analog subscriber connection 488. A fully digital connection is opened between digital subscriber 480 and a relay server 484 using a digital connection 482 such as ISDN, Switched-56, T1, or the like. Relay server 484 then communicates along a relay connection 486 with analog subscriber 490 using any means available such as a traditional modem or a system such as was shown in Figure 17. With appropriate flow-control methods, which are well known to those skilled in the art, it will appear to the digital subscriber that a digital connection has been opened to the analog-only subscriber. Such a connection can be used for any digital communication, such as voice, data, digital FAX, video, audio, etc. Note that it is also possible to incorporate relay server 484 into the actual digital telephone network 134 to provide apparent digital connectivity to analog subscribers transparently.

SCOPE

While the invention has been described in connection with, what is presently considered to be, the most practical and preferred embodiments, it is to be understood that the invention is not limited to the disclosed embodiments, but on the contrary, it is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims. For example, an equivalent training request can be accomplished by using the reverse channel in Figure 17. The reverse channel of Figure 17 also can provide other equivalent configurations for the control of information flow from decoder 156 to the encoder 150. However, in such a configuration, the present invention still provides the transfer of data between the data provider and consumer. In addition, compensation of a telephone line may be accomplished by other equivalent configurations, which are well known to those skilled in the art; equivalent training procedures may be used, different equalization methods may be utilized, and the system may be adapted to other central office equipment without departing from the scope of the invention. Therefore, persons of ordinary skill in this field are to understand that all such equivalent arrangements and modifications are to be included within the scope of the following claims.

APPENDIX-Example Pseudo-code Implementations

The following pseudo-code segments are provided to aid in understanding the various parts of the present invention. They should not be construed as complete or optimal implementations. Note that these codes illustrate the operation of the basic system described above, without any of the additional enhancements discussed. Although given as software code, the actual implementations may be as stored-program(s) used by a processor, as dedicated hardware, or as a combination of the two.

Example Implementation of decoder 156

```
/* Output begin training sync pattern */
for (i=0;i<syncPhaseLength;i++)
        Output maximum code value
for (i=0;i<syncPhaseLength;i++)
        Output minimum code value

/* Output training data */
for (i=0;i<trainRepeat;i++)
        for (j=0;j<trainLength;j++)
                Output training pattern element j

/* Output end training sync pattern */
for (i=0;i<syncPhaseLength;i++)
        Output minimum code value
for (i=0;i<syncPhaseLength;i++)
        Output maximum code value

sum=0
loop forever
        Read an input data byte
        Output data byte
        Convert byte to equivalent linear value, x
        sum += x;
        if (current output sample number is a multiple of dcPeriod)
                if (sum > 1.0)
                        recover = maximum code value
                else if (sum < -1.0)
                        recover = minimum code value
                else
                        recover = code value having linear value closest to -sum


                Output recover
                sum -= linear equivalent of recover
```

Example Implementation of clock synchronizer 260

Initialize filters array to be the impulse response of a low-pass

```
        filter with digital cutoff frequency PI/Nu.

Initialize lpfBuffer array to all zeroes.

snum = -lpfLen/2;
lpfPos = 0;

Loop forever
        Read an input sample into val
        /* Store value in circular buffer, lpfBuffer[] */
        lpfBuffer[lpfPos] = val;
        lpfPos = (lpfPos+1)%lpfLen;
        snum++;
        while (snum >= period)
                /* Extract an output from resampler at 'period' units after
                  the previous extracted sample */
                snum = snum - period;
                phase = (int)(snum*Nu);
                frac = snum*Nu-phase;

                /* Compute output from two adjacent phases of filter */
                                                    lpfOut1 = lpfOut2 = 0;
                for (i=0,p=lpfPos;i<lpfLen;i++,p=(p+1)%lpfLen)
                        lpfOut1 += lpfBuffer[p]*filters[i*Nu+phase];
                        lpfOut2 += lpfBuffer[p]*filters[i*Nu+phase+1];
                /* Interpolate */
                result = lpfOut1*(1-frac)+lpfOut2*frac;

                Write result as an output sample
```

Example Implementation of decoder 156

```
Loop forever
        Read a sample from clock synchronizer into 'samp'
        /* Put samp at the end of 'inBuffer' */
        inBuffer[inPos] = samp;
        inPos = (inPos+1)%inBufLen;

        /* Check if we are just finishing a sync pattern */
        if (last syncLength samples read are all negative and previous
```

```
        syncLength samples are all positive)
        inTraining = 1;
else if (last syncLength samples read are all positive and previous
syncLength samples are all negative)
        inTraining = 0;

/* Add sample to FFE buffer */
ffeBuffer[ffePos] = samp;
ffePos = (ffePos+1)%ffeLen;

/* Only need to compute output every second sample */
if (ffePos%2 == 0)
        /* Perform FFE equalization */
        ffeOut = DotProd(&ffeBuffer[ffePos],&ffeWts[0],ffeLen-
        ffePos);
        ffeOut += DotProd(&ffeBuffer[0],&ffeWts[ffeLen-
        ffePos],ffePos);

        /* Subtract FBE output */
        ffeOut -= fbeOut;

        /* Convert output to nearest code */
        codeOut = Linear2Code(ffeOut);

    if (inTraining)
            /* Use training pattern to calculate error */
            eEst = ffeOut - Code2Linear(train[tpos])
            tpos = (tpos+1)%trainLength;
    else
            /* Calculate decision feedback error */
            eEst = ffeOut-Code2Linear(codeOut);

    /* Update equalizers */
    for (i=0;i<ffeLen;i++)
            ffeWts[i] += ffeGain*eEst*ffeBuffer[(ffePos+i)%ffeLen];
    for (i=0;i<fbeLen;i++)
            fbeWts[i] += fbeGain*eEst*fbeBuffer[(fbePos+i)%fbeLen];
    /* Calculate derivative of output with respect to time */
    out[0] = out[1];
    out[1] = out[2];
```

```
out[2] = ffeOut;
deriv = (out[2]-out[0])/2;

/* Calculate phase error */
num *= pllPole;
denom *= pllPole;
num += prevEEst*deriv;
denom += deriv*deriv;
pdAdjust = num/denom;

/* Update resampler period (fed to clock synchronizer) */
period = midPeriod+pllGain*pdAdjust;

/* Save error estimate for next cycle */
prevEEst = eEst;

/* Compute next FBE output */
fbeBuffer[fbePos] = ffeOut;
fbePos = (fbePos+1)%fbeLen;
fbeOut = DotProd(&fbeBuffer[fbePos],&fbeWts[0],fbeLen- fbePos);
fbeOut += DotProd(&fbeBuffer[0],&fbeWts[fbeLen- fbePos],fbePos);

/* Output a sample (delayed) if we are active */
if (outputting)
        if (oSampNum>0 && (oSampNum%dcPeriod) != 0)
                Output outBuffer[outBufPos]
        oSampNum++;

/* Store new sample in output buffer */
outBuffer[outBufPos] = codeOut;
outBufPos = (outBufPos+1)%outBufLen;

/* Check if sync in buffer and set outputting accordingly */          if
(last syncLength/2 samples placed in outBuffer are negative
                and previous syncLength/2 samples are positive)
        outputting = 0;
else if (last syncLength/2 samples placed in outBuffer are
        negative and prev. syncLength/2 samples are positive)
outputting = 1;
        oSampNum = -syncLength + 1;
```

```
#include <stdio.h>
#include <bstring.h>
#include <math.h>
#include <assert.h>

#include "defs.h"
#include "trace.h"
#include "encode.h"
#include "decode.h"
#include "ulaw.h"
#include "nr.h"
#include "remapping.h"

/* Return 1 if zero suppression present */
static int ZSuppress(const double codemap[NBRPHASES][256],
                     const int ncodemap[NBRPHASES][256])
{
    double cw0 = 0.0;
    int cw0cnt = 0;
    int j;

    for (j=0;j<NBRPHASES;j++) {
        /* Average codemap of codeword 0 */
        cw0 += codemap[j][0];
        cw0cnt += ncodemap[j][0];
    }
    if (cw0cnt == 0) {
        return 0;
    }
    cw0 /= cw0cnt;

    TRACED(cw0);

    /* Check for zero-suppression */
    return fabs(cw0-Mu2Linear(0)) > fabs(cw0-Mu2Linear(2));
}

/* Reset dp->remap based on observed linear values for each code for each phase\

   Use mean observations for remap if 'usemap' is nonzero.  Otherwise, determine
   network conditions (zero-suppression, bit-robbing) and set remap to expected
   linear values from Mu->Linear map
   */
int LineCheck(DecParams *dp, const double codemap[NBRPHASES][256],
              const double codemap2[NBRPHASES][256],
              const int ncodemap[NBRPHASES][256], int usemap)
{
    int i,j,k,g,sign;
    /* Intermediate sums */
    double srr[NBRPHASES][NMAPS],src[NBRPHASES][NMAPS],sr[NBRPHASES][NMAPS];
    double scc[NBRPHASES],sc[NBRPHASES];
    int sn[NBRPHASES];
    /* Best values of a,b for each of the phases computed independently */
    double besta[NBRPHASES], bestb[NBRPHASES];
    /* Best value of the map (k) indexed by which besta,bestb are used and then\
 by which phase */
    int mink[NBRPHASES][NBRPHASES];
    /* Total error using the each of the elements of besta,bestb */
    double totalerr[NBRPHASES];
    /* Best gain index */
    int bestg;
    /* Overall sums */
    double allsrr=0,allsrc=0,allsr=0,allscc=0,allsc=0;
    int allsn=0;
    /* Overall gain and offset */
    double alla,allb;

    /* Calculate RMS Error for each possible remapping */

    /* We need to find the optimal map for each phase under the condition that
the gain is
       the same for all phases.  That is done in steps:
       1.  Find the best map and gain for each phase independent of the o    p
hases.
       2.  For each of the 6 gains found in step 1, find the best maps for each
 of the 6 phases.
       3.  Choose the best of the 6 sets from part 2.
       4.  Recalculate the gain using the best mapping set.
       */

    /* Step 1 - Find the best gains for each phase and map */
    for (j=0;j<NBRPHASES;j++) {
        double rmserr[NMAPS], a[NMAPS], b[NMAPS];

        fprintf(stderr,"Phase %d ", j);
        scc[j]=0;
        sc[j]=0;
        sn[j]=0;
        for (i=0;i<256;i++)
            /* Don't look at the outer codewords - zero suppression or
               saturation may distort them */
            if ((i&0x7c) != 0) {
                int n=ncodemap[j][i];
                scc[j] += codemap2[j][i];
                sc[j] += codemap[j][i];
                sn[j] += n;
            }
        for (k=0;k<NMAPS;k++) {
            srr[j][k]=0;
            sr[j][k]=0;
            src[j][k]=0;
            for (i=0;i<256;i++)
                if ((i&0x7c) != 0) {
                    int n=ncodemap[j][i];
                    double r = Remap(k,i);
                    double cn = codemap[j][i];
                    srr[j][k] += n*r*r;
                    sr[j][k] += n*r;
                    src[j][k] += r*cn;
                }
            a[k] = (src[j][k]*sn[j]-sc[j]*sr[j][k])/(srr[j][k]*sn[j]-sr[j][k]*s\
r[j][k]);
            b[k] = (sc[j]-a[k]*sr[j][k])/sn[j];
            b[k] = (sc[j]-a[k]*sr[j][k])/sn[j];
            rmserr[k] = sqrt((a[k]*a[k]*srr[j][k]-2*a[k]*src[j][k]-2*b[k]*   ]\
+scc[j]+                      2*a[k]*b[k]*sr[j][k]+b[k]*b[k]*sn[j])/sn[...;
            fprintf(stderr,"%s=%.0f ",MapID[k],rmserr[k]*8192);
        }
        fprintf(stderr,"\n");
        mink[j][j] = 0;
        for (k=1;k<NMAPS;k++)
            if (rmserr[mink[j][j]] > rmserr[k])
                mink[j][j]=k;
        besta[j] = a[mink[j][j]];
        bestb[j] = b[mink[j][j]];
        fprintf(stderr,"best=%s, err*8192=%.0f, a=%f b=%g\n", MapID[mink[j][j]]\
 rmserr[mink[j][j]]*8192,
                besta[j], bestb[j]);
    }
```

```
    /* Step 2 - Find best maps for each of the 6 phases using each of the "best\
" gains */
    for (g=0;g<NBRPHASES;g++) {
        double a = besta[g];
        double b = bestb[g];
        totalerr[g] = 0;
        for (j=0;j<NBRPHASES;j++)
            /* Already have done the case where g==j */
            if (j!= g) {
                double minrmserr = 1e38;
                for (k=0;k<NMAPS;k++) {
                    double rmserr = sqrt((a*a*srr[j][k]-2*a*src[j][k]-2*b*sc[j]\
+scc[j]+
                                        2*a*b*sr[j][k]+b*b*sn[j])/sn[j]);
                    if (rmserr < minrmserr) {
                        minrmserr = rmserr;
                        mink[g][j] = k;
                    }
                }
                totalerr[g] += minrmserr;
            }
    }

    /* Step 3 - Choose best of the 6 gains */
    bestg = 0;
    for (g=1;g<NBRPHASES;g++)
        if (totalerr[g] < totalerr[bestg])
            bestg = g;

    /* Step 4 - Set the maps for each phase */
    for (j=0;j<NBRPHASES;j++) {
        if (dp->mapping[j] != mink[bestg][j])
            fprintf(stderr,"Phase %d went from %s to %s.\n",j,MapID(dp->mapping\
[j]),MapID(mink[bestg][j]));
        dp->mapping[j] = mink[bestg][j];
    }

    /* Step 5 - recompute overall gain and offset */
    for (j=0;j<NBRPHASES;j++) {
        allsrr += srr[j][dp->mapping[j]];
        allsrc += src[j][dp->mapping[j]];
        allsr += sr[j][dp->mapping[j]];
        allscc += scc[j];
        allsc += sc[j];
        allsn += sn[j];
    }
    alla = (allsrc*allsn-allsr*allsc)/(allsrr*allsn-allsr*allsr);
    allb = (allsc-alla*allsr)/allsn;
    TRACED(allb);
    dp->codecDeviation = sqrt((alla*alla*allsrr-2*alla*allsrc-2*allb*allsc+alls\
cc+
                              2*alla*allb*allsr+allb*allb*allsn)/allsn);
    fprintf(stderr,"a=%f\tb*8192=%.0f\tdev*8192=%.0f\n",alla,allb*8192,dp->code\
cDeviation*8192);

    /* Set remapping */
    for (j=0;j<NBRPHASES;j++)
        for (i=0;i<256;i++)
            dp->remap[j][i] = Remap(dp->mapping[j],i)*alla;

    /* Determine zero-suppression (assume same for all phases) */
    for (sign=0;sign<=0x80;sign+=0x80)
        for (i=0;i<3;i++) {
            float se2[4]={0,0,0,0};
            int bestk= -1;
            for (j=0;j<NBRPHASES;j++) {
                for (k=i;k<4;k++) {
                    double e;
                    e = ncodemap[j][i|sign]*(dp->remap[j][k|sign]+allb)-codema\
p[j][i|sign];
                    se2[k] += e*e;
                }
            }
            bestk=i;
            for (k=i+1;k<4;k++)
                if (se2[k] < se2[bestk])
                    bestk=k;
            if (i!=bestk) {
                fprintf(stderr,"Zero Suppression: %d->%d\n",i|sign,bestk|sign);
                for (j=0;j<NBRPHASES;j++)
                    dp->remap[j][i|sign]=dp->remap[j][bestk|sign];
            }
        }

    /* Verify computations */
    for (j=0;j<NBRPHASES;j++) {
        double se2 = 0;
        double se = 0;
        int c2 = 0;
        for (i=0;i<256;i++) {
            double e;
            e = (dp->remap[j][i]+allb)-codemap[j][i]/ncodemap[j][i];
            c2++;
            se2 += e*e;
            se += e;
        }
        se2 = sqrt(se2/c2);
        se = se/c2;
        fprintf(stderr,"Phase %d: %s e2*8192=%.0f, e*8192=%.0f\n",j,MapID(dp->m\
apping[j]),se2*8192,se*8192);
    }

    trace_float_array_nc("remap",&dp->remap[0][0],NBRPHASES*256);
    trace_cmd("remap=reshape(remap,256,length(remap(:))/256);\n");

    if (usemap) {
        /* Use observed values for midpoints */
        for (j=0;j<NBRPHASES;j++) {
            /* Remove any DC components, otherwise DC drift will cause the code\
map to shift
               and get worse each time it is updated */
            float sum=0;
            int nsum=0;
            for (i=0;i<256;i++) {
                if (ncodemap[j][i] > 10)
                    dp->remap[j][i] = codemap[j][i]/ncodemap[j][i]  -allb ;
                else {
                    fprintf(stderr,"Only %d examples for codemap[%d][%d],    g\
 remap, %.3f, instead of %.3f\n",
                            ncodemap[j][i],j,i,dp->remap[j][i],codemap[j][i]/nc\
odemap[j][i]  -allb );
                }
                /* Compute mean level (skip outer levels in case of zero suppre\
ss or sat. */
                if ((i&0x7c) != 0) {
                    sum+=dp->remap[j][i];
                    nsum++;
                }
            }
            /* Remove DC from each codemap individually */
            sum /= nsum;
            for (i=0;i<256;i++)
```

```
            dp->remap[j][i] -= sum;
        }
    }
    return 1;  /* Always changed (for now) */
}

void ShowRemap(const double codemap[NBRPHASES][256],
               const int ncodemap[NBRPHASES][256])
{
    int i,j;
    for (j=0;j<NBRPHASES;j++) {
        float s,s1;
        float minv=1e38,mins=0;
        float e2=0;
        int n=0;

        /* Display best fit to a scaled version of mulaw */
        for (s=0.45;s<1.1;s+=.001) {
            double v=0;
            for (i=0;i<20;i++) {
                double e=Mu2Linear(Linear2Mu(s*codemap[j][i]/ncodemap[j][i]))
                    *ncodemap[j][i]/s-codemap[j][i];
                v+=e*e;
            }
            if (v < minv) {
                minv=v;
                mins=s;
            }
        }
        s1=mins;
        for (s=s1-.01;s<s1+.01;s+=.0001) {
            double v=0;
            for (i=0;i<20;i++) {
                double e=Mu2Linear(Linear2Mu(s*codemap[j][i]/ncodemap[j][i]))
                    *ncodemap[j][i]/s-codemap[j][i];
                v+=e*e;
            }
            if (v < minv) {
                minv=v;
                mins=s;
            }
        }

        for (i=0;i<256;i++) {
            double e=Mu2Linear(Linear2Mu(mins*codemap[j][i]/ncodemap[j][i]))
                /mins-codemap[j][i]/ncodemap[j][i];
            e2+=e*e;
            n++;
        }

        fprintf(stderr,"Remap %d %.4f e=%.0f ",j,mins,sqrt(e2/n)*8192);
        for (i=0;i<50;i++) {
            int v=Linear2Mu(mins*codemap[j][i]/ncodemap[j][i]);
            char c;
            if (v<10)
                c='0'+v;
            else if (v<36)
                c='A'+v-10;
            else if (v<62)
                c='a'+v-36;
            else
                c='+';
            fprintf(stderr,"%c",c);
        }
        fprintf(stderr,"\n");
    }
```

```c
#include <assert.h>
#include <string.h>
#include <math.h>
#include "defs.h"
#include "ulaw.h"
#include "remapping.h"
#include "trace.h"

static float maps[NMAPS][256];
const char *MapID[]= {
    "d","h","s",
    "1","1h","1s","h1","h1h","h1s",
    "2","2h","2s",
    "3","3h","3s","h3","h3h","h3s",
    "4","4h","4s","h4","h4h","h4s",
    "5","5h","5s","h5","h5h","h5s",
};

/* Apply a 6dB digital pad to the input value */
static byte DPad(byte in,int typ)
{
    switch (typ) {
    case 2:
        /* It looks like some lines pad the values +/- 32 a little differently \
*/
        if ((in&0x7f)==32)
            return 48|(in&0x80);
    case 1:
        return Linear2Mu(Mu2Linear(in)/2);
    case 3:
        /* Another mapping that treats values 32-48 differently */
        return Linear2Mu(Mu2Linear(in)/2.015);
    case 4:
        /* 3dB Pad */
        if ((in&0x7f)==25)
            return 31|(in&0x80);
        return Linear2Mu(Mu2Linear(in)/M_SQRT2);
    case 5:
        /* Another 3dB Pad */
        return Linear2Mu(Mu2Linear(in)/1.391);
    default:
        abort();
    }
    abort();
}

static float DoMapping(byte in, const char *mapstr)
{
    byte out=in;
    for (;*mapstr;mapstr++)
        switch (*mapstr) {
        case 'd':
            break;
        case 'h':
            out=out|1;
            break;
        case 's':
            assert(mapstr[1]==0);
            return (Mu2Linear(out&0xfe)+Mu2Linear(out|1))/2;
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
            out=DPad(out,(*mapstr)-'0');
            break;
        case 'z':
            if (out==0)
                out=2;
            break;
        case 'Z':
            if (out==0 || out==1)
                out=2;
            if (out==128 || out==129)
                out=130;
            break;
        default:
            abort();
        }
    return Mu2Linear(out);
}

/* Create Mappings */
void MakeMaps(void)
{
    int j,i;
    byte pad[256];
    assert(NMAPS==sizeof(MapID)/sizeof(MapID[0]));
    for (j=0;j<256;j++)
        for (i=0;i<NMAPS;i++)
            maps[i][j] = DoMapping(j,MapID[i]);

    /* Dump maps */
    for (j=1;j<=4;j++) {
        for (i=0;i<256;i++)
            pad[i] = DPad(i,j);
        trace_byte_array_nc("pad",pad,256);
    }
    trace_reshape("pad",256,4);
    trace_close("pad");
}

/* Using given mapping, convert a mulaw code to the expected float value */
float Remap(int mapping, byte code)
{
    return maps[mapping][code];
}

const char *MapStr(int n, int remapping[])
{
    static char result[100];
    int i,primeph=0,ph;
    double primesum = 1e308;

    for (ph=0;ph<n;ph++) {
        double robsum = remapping[ph];;
        for (i=1;i<n;i++)
            robsum = robsum*NMAPS + remapping[(i+ph)%n];
        if (robsum < primesum) {
            primesum = robsum;
            primeph = ph;
        }
    }

    result[0] = 0;
    for (i=0;i<n;i++) {
        strcat(result,MapID[remapping[(i+primeph)%n]]);
        strcat(result," ");
    }
    sprintf(result,"%s %d",result,primeph);
    return result;
}
```

WHAT IS CLAIMED IS:

1. In a communication system that includes an encoder and a decoder, wherein the encoder has a digital connection to a digital portion of a telephone network and the digital portion of the telephone network is connected by an analog loop to the decoder, a training method comprises the steps of:

transmitting M repetitions of N codewords from the encoder to the decoder, where M and N are integers that are known to the decoder and said codewords correspond to PCM codes utilized by the digital portion of the telephone network;

the decoder; sampling a received sequence of analog voltage levels corresponding to said transmitted codewords;

storing a value associated with each sample;

analyzing the stored values; and

adjusting a parameter of the decoder in accordance with the analysis.

2. A training method as claimed in claim 1, wherein the N codewords are randomly selected from a subset of two PCM codewords, and wherein the subset of two PCM codewords are negatives of each other under a.μ-law companing rule implemented by the telephone network.

3. A training method as claimed in claim 2, wherein each of said two PCM codewords is selected N/2 times.

4. A training method as claimed in claim 1, wherein said received sequence is sampled at twice the clock rate of the digital portion of the telephone network.

5. A training method as claimed in claim 1, wherein said received sequence is sampled at a rate of 16,000 samples/second.

6. A training method as claimed in claim 1, wherein said steps of analyzing the stored values comprises the step of determining a rate at which the M repetitions occur.

7. A training method as claimed in claim 6, wherein said steps of analyzing further comprises the steps of determining a sample period associated with the rate at which the M repetition occurs; comparing the determined period to N/8000 seconds; and adjusting a sampling rate clock in the decoder in accordance with the comparison.

8. A training method as claimed in claim 7, further comprising the step of resampling the stored values using the adjusted sampling rate clock.

9. A training method as claimed in claim 8, further comprising the steps of repeating the steps of claim 7 on the resampled values.

10. A training method as claimed in claim 1, wherein the steps of analyzing the stored values comprises the step of measuring the variance of the M repetitions to thereby measure a noise estimate.

11. A training method as claimed in claim 1, wherein the step of analyzing the stored values comprises the step of calculating an average received signal level from the M repetitions.

12. A training method as claimed in claim 1, wherein said parameter is associated with an equalizer in said decoder.

13. A training method as claimed in claim 12, wherein said parameter is adjusted in accordance with at least one of a predetermined sequence stored at the decoder, a measure of average received signal level, and a noise estimate.

14. A training method as claimed in claim 12, wherein said step of analyzing comprises calculating a mean-squared error estimate.

15. A training method as claimed in claim 14, wherein said parameter is adjusted to achieve an equalizer having a minimum mean-squared error.

16. A training method as claimed in claim 15, further comprising the step of transmitting said parameter back to said encoder.

17. A training method as claimed in claim 16, further comprising the step of, at said encoder, choosing a codeword set for data transmission based upon said parameter.

18. A training method as claimed in claim 1, further comprising the step of updating said training based upon a measure of deviation between a compensated signal and a nearest codeword.

19. A training method as claimed in claim 18, further comprising the step of continuously updating a sampling rate clock and a parameter associated with an equalizer in accordance with said measure of deviation.

20. A training method as claimed in claim 19, further comprising the step of retraining when said measure of deviation exceeds a predetermined threshold.

21. In a communication system that includes an encoder and a decoder, wherein the encoder has a digital connection to a digital portion of a telephone network and the digital portion of the telephone network is connected by an analog loop to the decoder, wherein the analog loop converts digital information from the network into analog voltages for transmission to the decoder, a method of detecting the presence of robbed-bit signaling within the digital portion of the telephone network comprising the steps of:

sending a predetermined pattern of PCM codewords from the encoder to the digital portion of the telephone network;

receiving said predetermined pattern of PCM codewords in the form of analog voltages at the decoder;

determining an error measurement at the decoder in response to said received analog voltages; and

determining whether robbed-bit signaling is present based upon said error measurements.

22. A method of detecting the presence of robbed-bit signaling as in claim 21 wherein the error measurement determining step includes comparing said received analog voltages to predetermined reference voltages.

23. A method of detecting the presence of robbed-bit signaling as in claim 21 wherein said predetermined pattern is partitioned into M frames, each of said M frames having N time slots.

24. A method of detecting the presence of robbed-bit signaling as in claim 23 wherein said error measurement step comprises forming N average error measurements corresponding to said N time slots.

25. A method of detecting the presence of robbed-bit signaling as in claim 24 wherein each of said N average error measurements is formed by averaging every Nth error measurement.

26. A method of detecting the presence of robbed-bit signaling as in claim 23 where N=6.

27. A method of detecting the presence of robbed-bit signaling as in claim 21 wherein said robbed-bit signaling determining step further comprises the steps of:

determining if the magnitudes of said error measurements are cyclically repetitive; and

when said error measurements are cyclically repetitive, determining whether said cyclically repetitive error measurements result from robbed-bit signaling.

28. In a communication system that includes an encoder and a decoder, wherein the encoder has a digital connection to a digital portion of a telephone network and the digital portion of the telephone network is connected by an analog loop to the decoder, a method of selecting a PCM codeword set comprising the steps of:

transmitting a sequence of PCM codewords from the encoder;

receiving said sequence as a sequence of analog voltages at the decoder;

determining a noise level at the decoder in response to said received sequence of analog voltages;

selecting a minimum separation value in response to said noise level determination; and

selecting a set of PCM codewords such that no two PCM codewords within said set are separated by less than said selected minimum separation value.

29. A method of selecting a PCM codeword set of claim 28 wherein said set selecting step further comprises the step of selecting a second a set of PCM codewords, wherein PCM codewords from said second set are used during bit-robbed time slots.

30. A method of selecting a PCM codeword set of claim 28 wherein said encoding step is performed by a convolutional encoder.

31. A method of selecting a PCM codeword set of claim 28 wherein said encoding step is performed by a block encoder.

32. A method of selecting a PCM codeword set of claim 28 wherein said encoding step is performed by a trellis encoder.

33. A method of selecting a PCM codeword set of claim 28 wherein said noise level determining step further comprises the step of measuring the variance of the received sequence of analog voltages.

34. In a communication system that includes an encoder and a decoder, wherein the encoder has a digital connection to a digital portion of a telephone network and the digital portion of the telephone network is connected by an analog loop to the decoder, a method of conveying information from the encoder to the decoder comprising the steps of:

transmitting a sequence of PCM codewords from the encoder;

receiving said sequence as a sequence of analog voltages at the decoder;

determining a noise level at the decoder in response to said received sequence of analog voltages;

selecting a minimum separation value in response to said noise level determination;

selecting a set of PCM codewords such that no two PCM codewords within said set are separated by less than said selected minimum separation value; and

encoding a sequence of information bits to a sequence of PCM codewords from said selected codeword set to convey information from said encoder to said decoder.

35. A method of conveying information of claim 34 wherein said set selecting step further comprises the step of selecting a second a set of PCM codewords, wherein PCM codewords from said second set are used during bit-robbed time slots.

36. A method of conveying information of claim 34 wherein said encoding step is performed by a convolutional encoder.

37. A method of conveying information of claim 34 wherein said encoding step is performed by a block encoder.

38. A method of conveying information of claim 34 wherein said encoding step is performed by a trellis encoder.

39. A method of conveying information of claim 34 wherein said noise level determining step further comprises the step of measuring the variance of the received sequence of analog voltages.

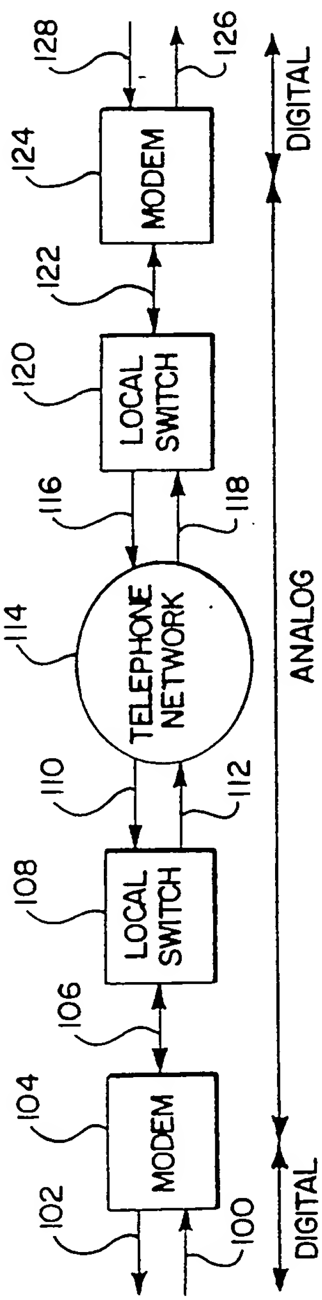


FIG. 1
PRIOR ART

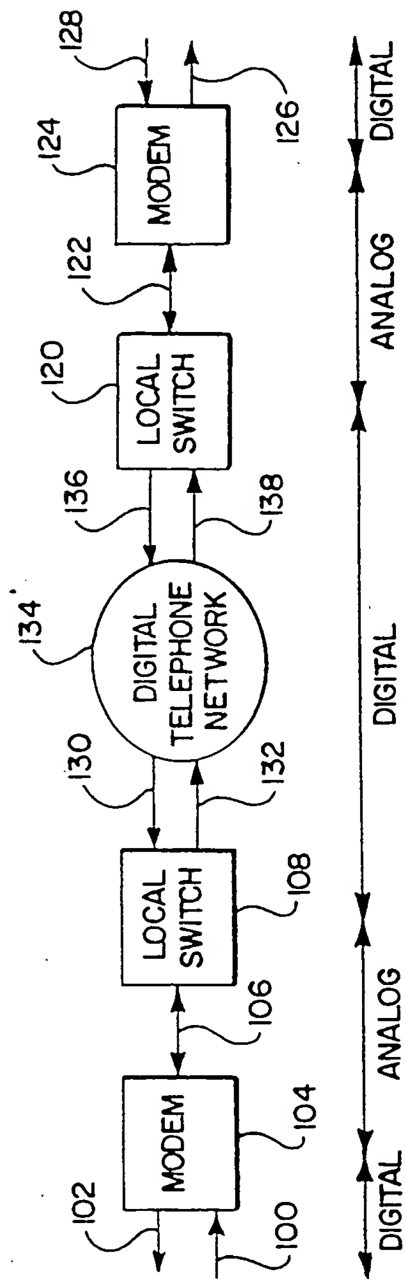


FIG. 2
PRIOR ART

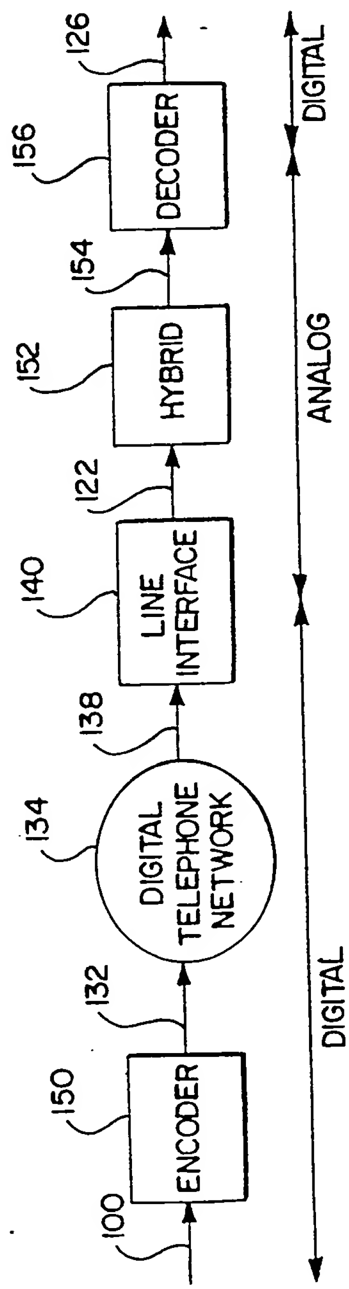

100
ENCODER
150
132
DIGITAL TELEPHONE NETWORK
134
138
LINE INTERFACE
140
122
HYBRID
152
154
DECODER
156
126
DIGITAL
ANALOG
DIGITAL

FIG. 3

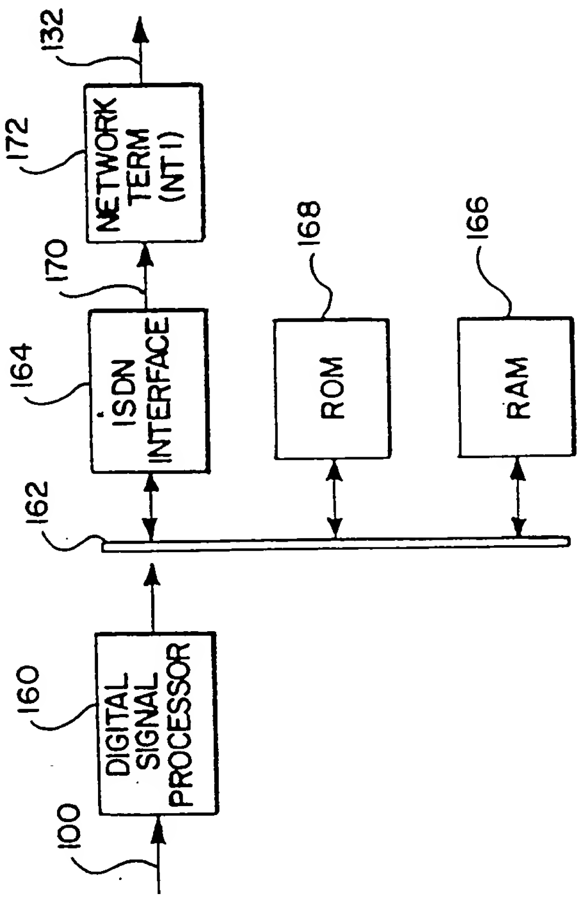

100
DIGITAL SIGNAL PROCESSOR
160
162
ISDN INTERFACE
164
170
NETWORK TERM (NT1)
172
132
ROM
168
RAM
166

FIG. 4

FIG. 5

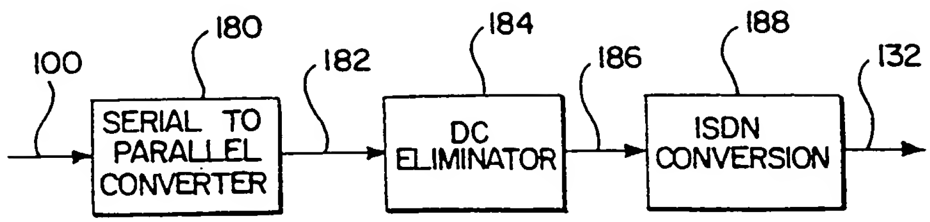
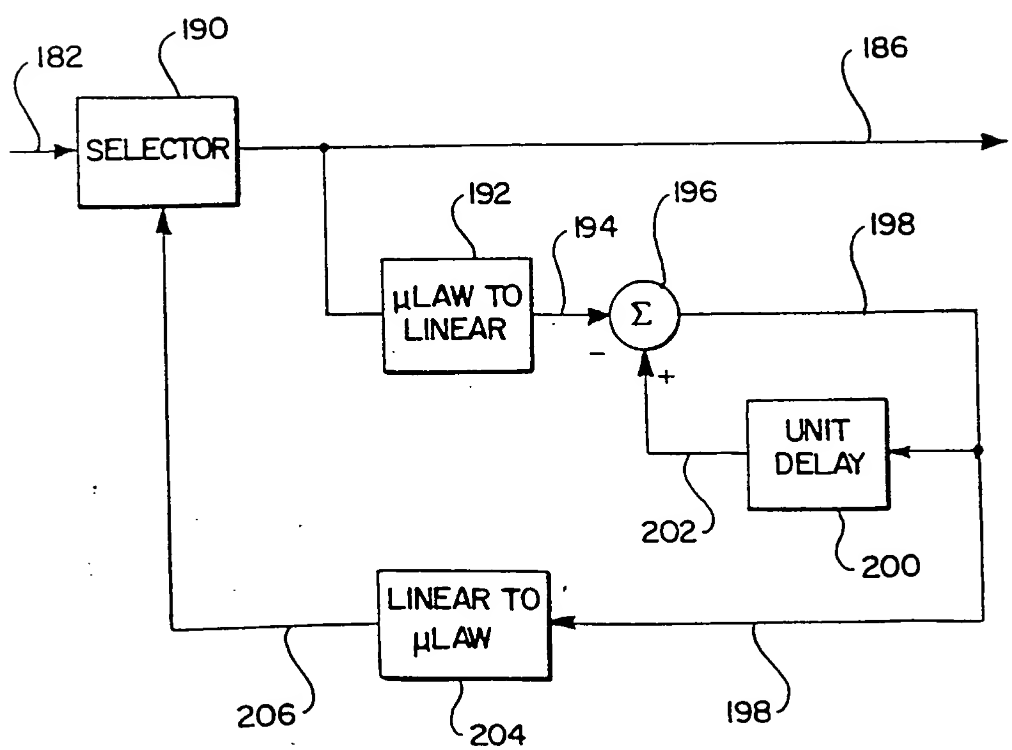

100
180
SERIAL TO PARALLEL CONVERTER
182
184
DC ELIMINATOR
186
188
ISDN CONVERSION
132

FIG. 6


182
190
SELECTOR
186
192
μLAW TO LINEAR
194
196
Σ
198
UNIT DELAY
202
200
LINEAR TO μLAW
206
204
198

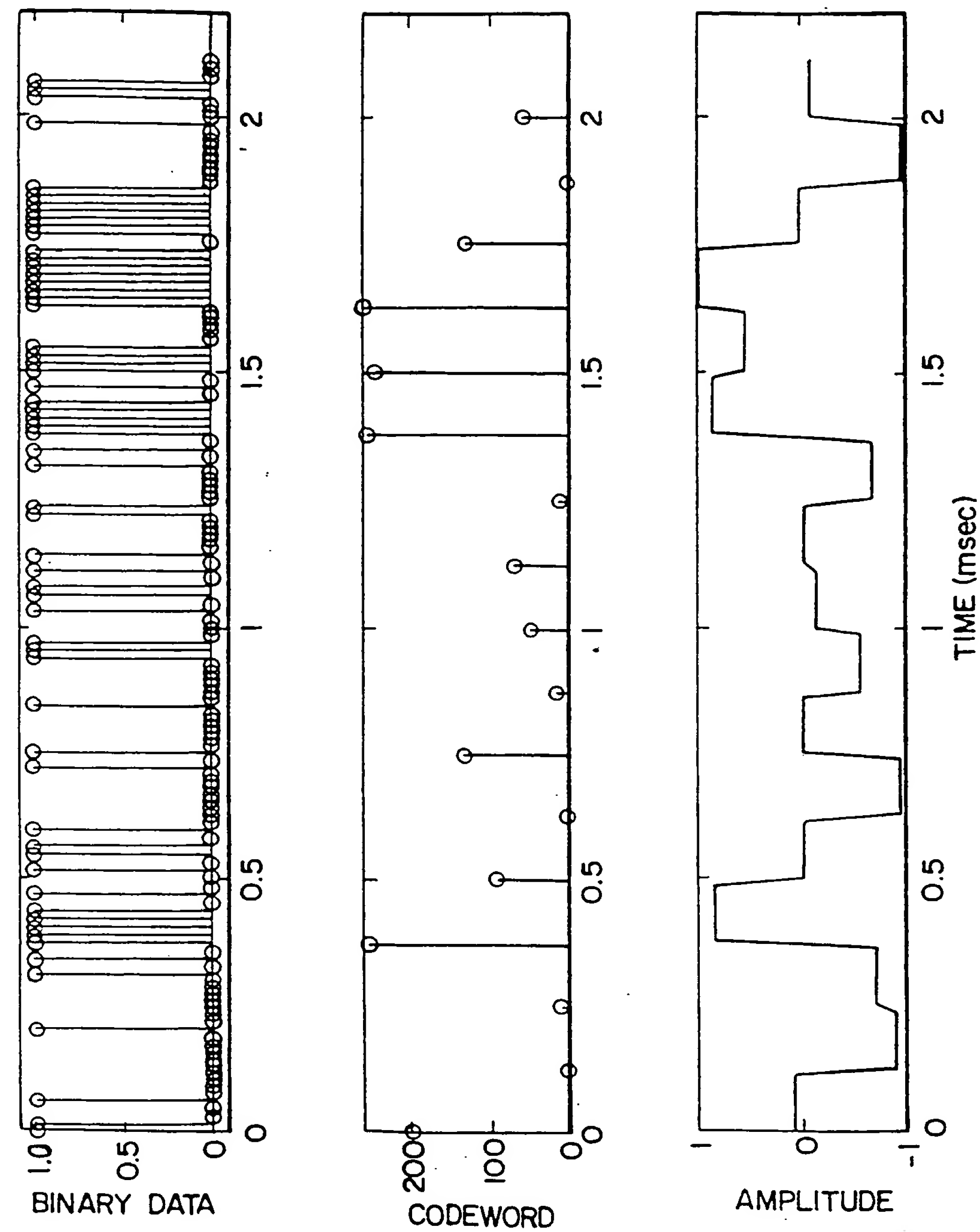


FIG. 7A

FIG. 7B

FIG. 7C

FIG. 8

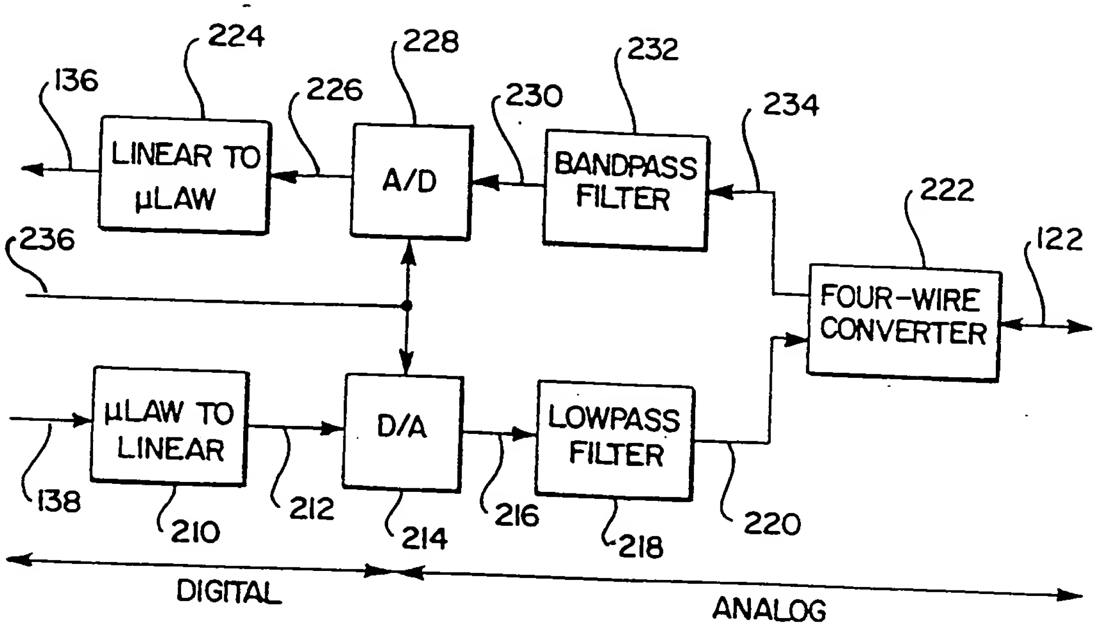

224
228
232
136
226
230
234
LINEAR TO
μLAW
A/D
BANDPASS
FILTER
222
236
122
FOUR-WIRE
CONVERTER
μLAW TO
LINEAR
D/A
LOWPASS
FILTER
138
210
212
214
216
218
220
DIGITAL
ANALOG

FIG. 9

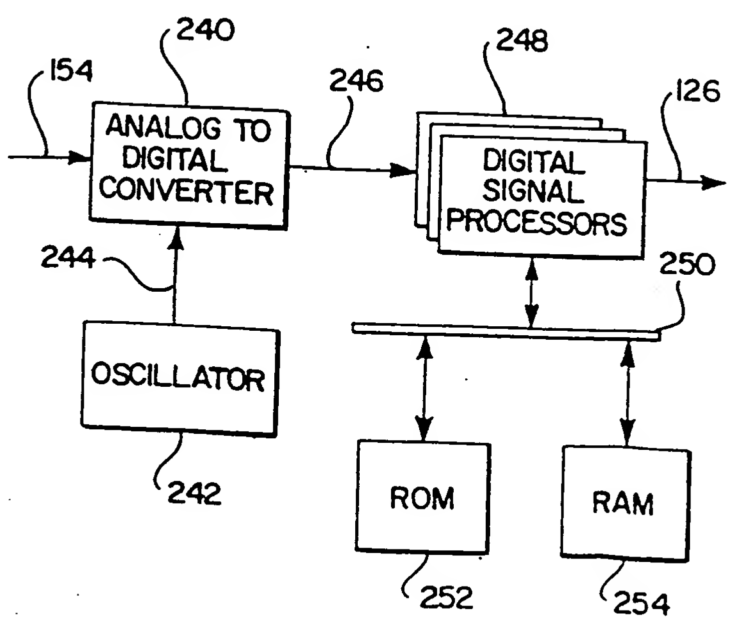

240
248
154
246
126
ANALOG TO
DIGITAL
CONVERTER
DIGITAL
SIGNAL
PROCESSORS
244
250
OSCILLATOR
242
ROM
RAM
252
254

FIG. 10

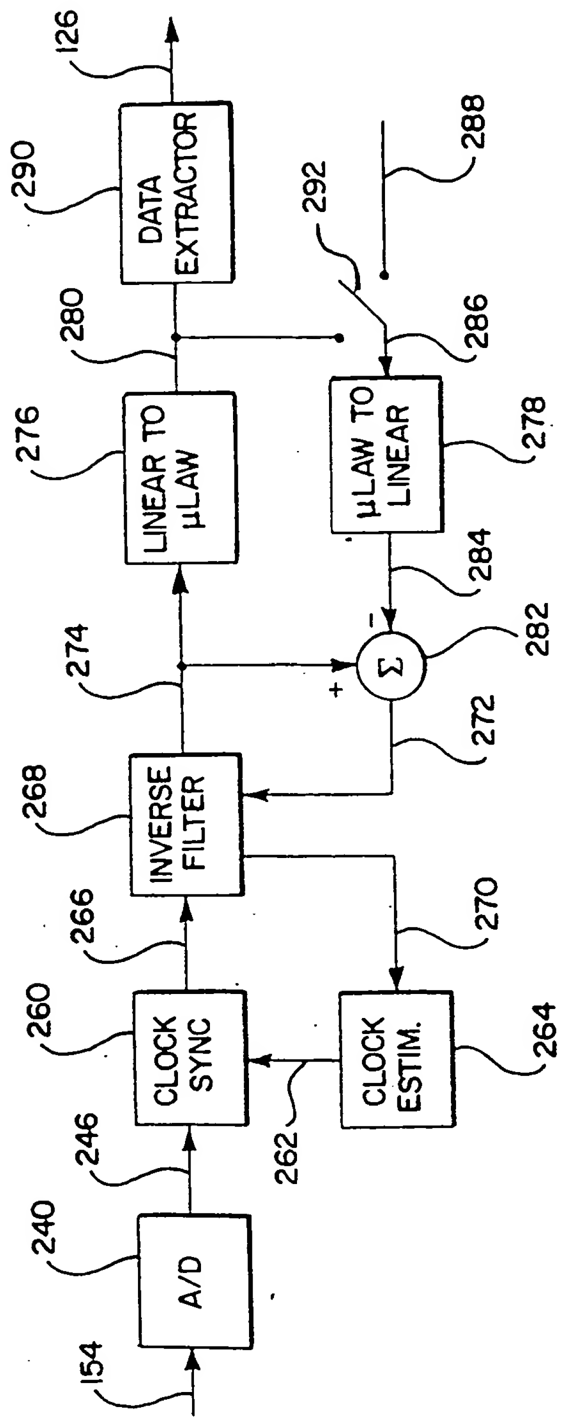

A/D
CLOCK SYNC
CLOCK ESTIM.
INVERSE FILTER
LINEAR TO µLAW
µLAW TO LINEAR
DATA EXTRACTOR
Σ
+
-
154
240
246
260
262
264
266
268
270
272
274
276
278
280
282
284
286
288
290
292
126

FIG. 11A

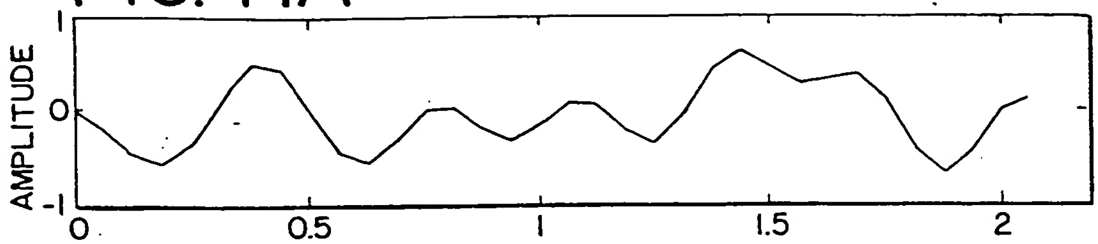


FIG. 11B

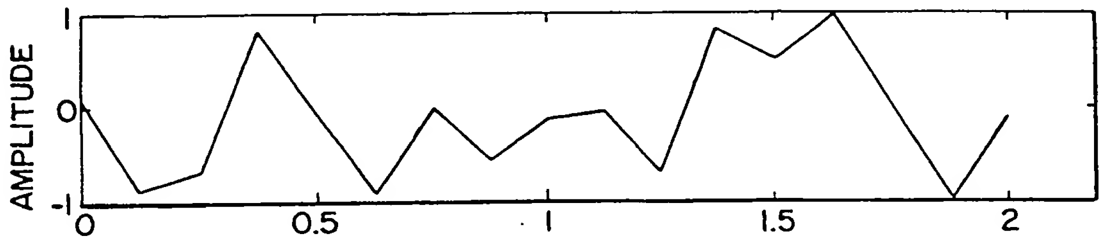


FIG. 11C

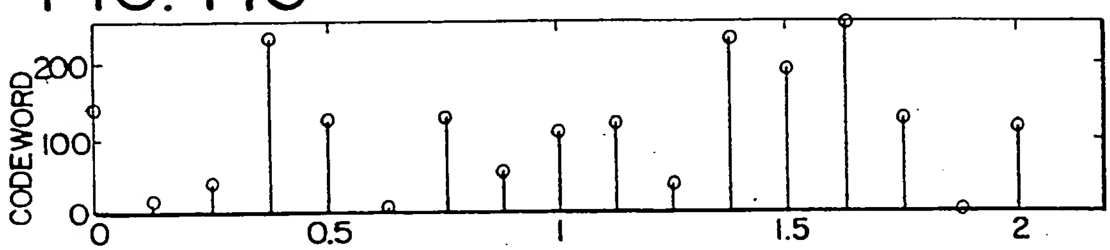


FIG. 11D

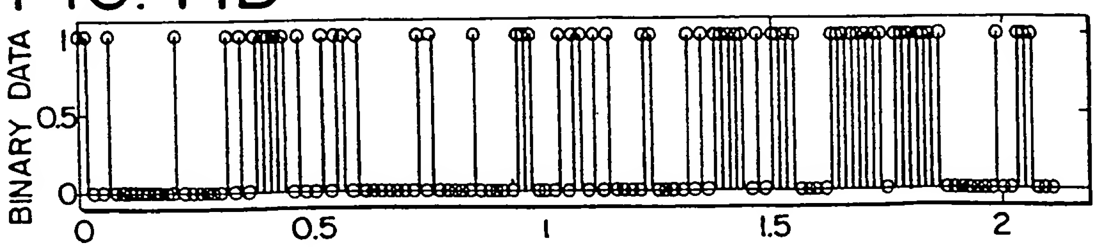


FIG. 11E

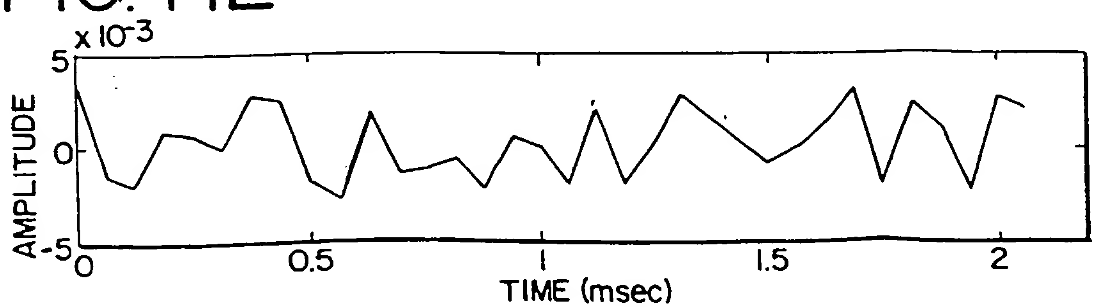


FIG. 12

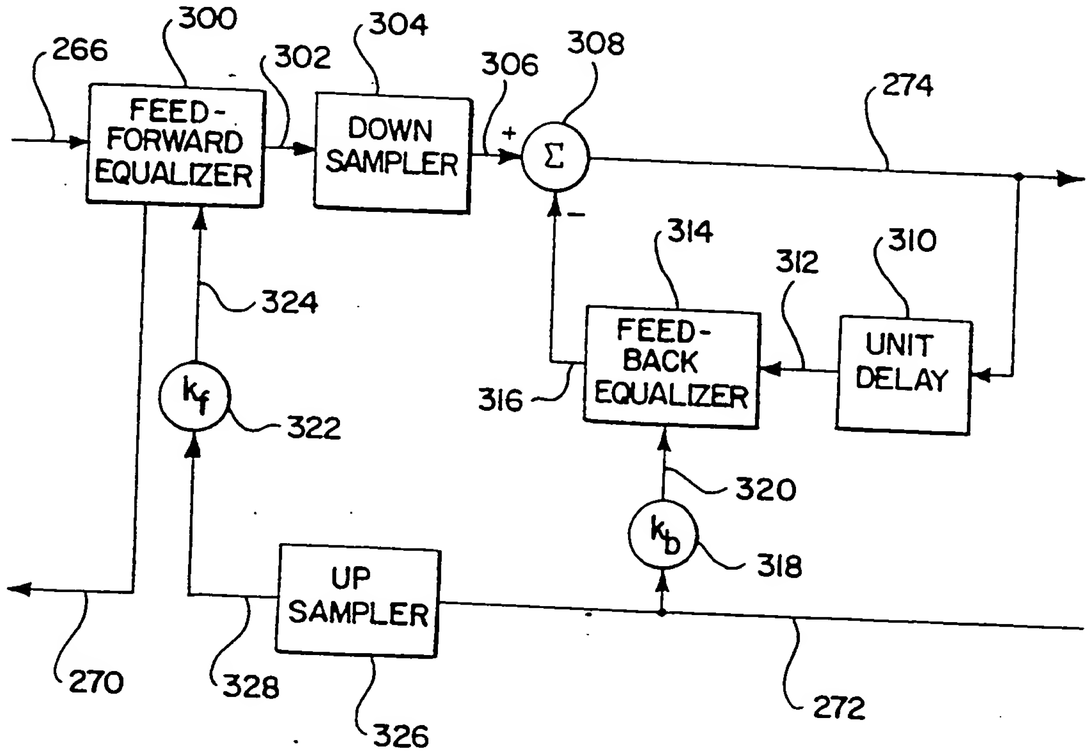

266
300
FEED-FORWARD EQUALIZER
302
304
DOWN SAMPLER
306
+
308
Σ
274
–
314
312
310
324
FEED-BACK EQUALIZER
UNIT DELAY
k_f
322
316
320
k_b
318
UP SAMPLER
270
328
326
272

FIG. 13

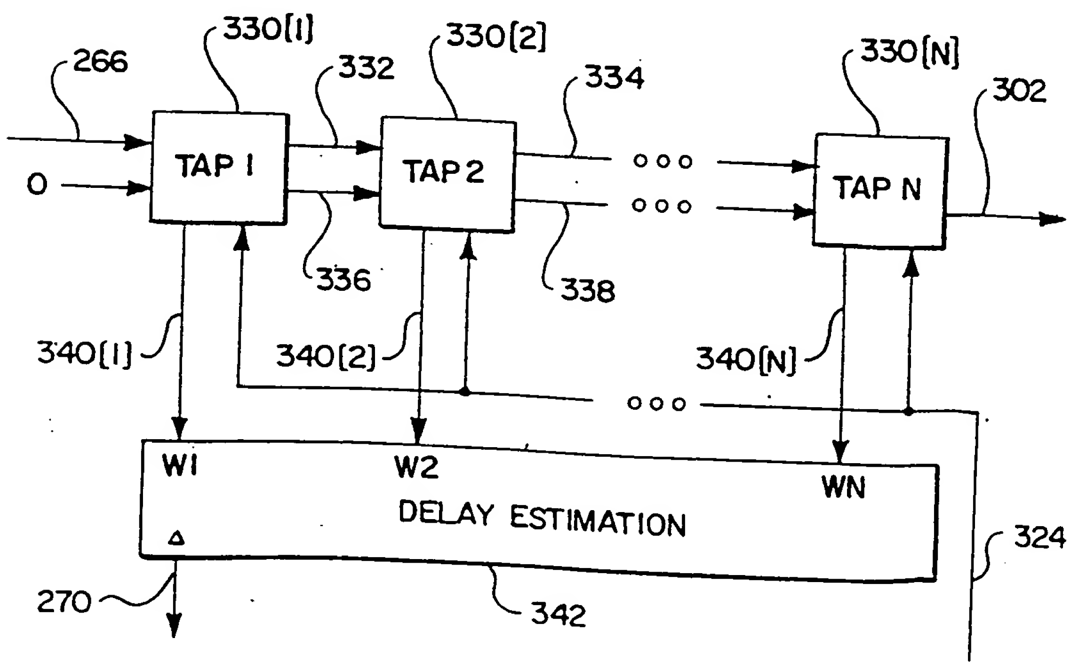

266
330[1]
332
330[2]
334
330[N]
302
TAP 1
TAP 2
TAP N
0
336
338
340[1]
340[2]
340[N]
W1
W2
WN
DELAY ESTIMATION
Δ
324
270
342

FIG. 14

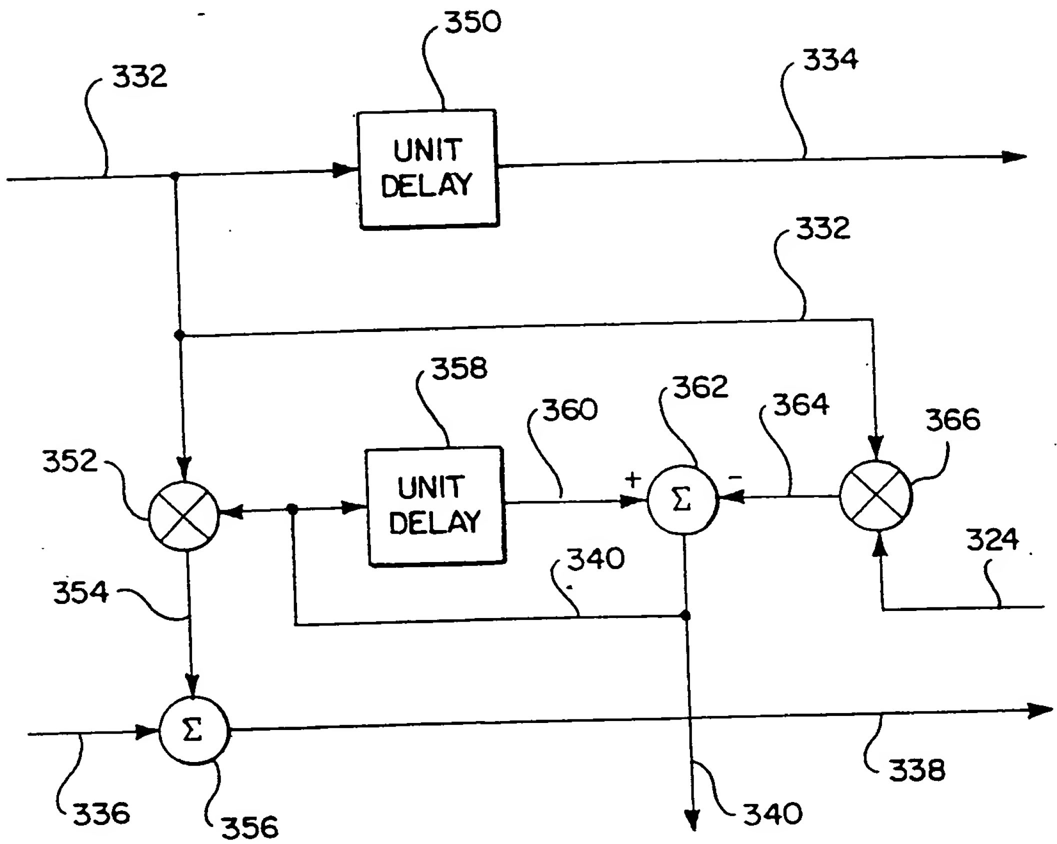
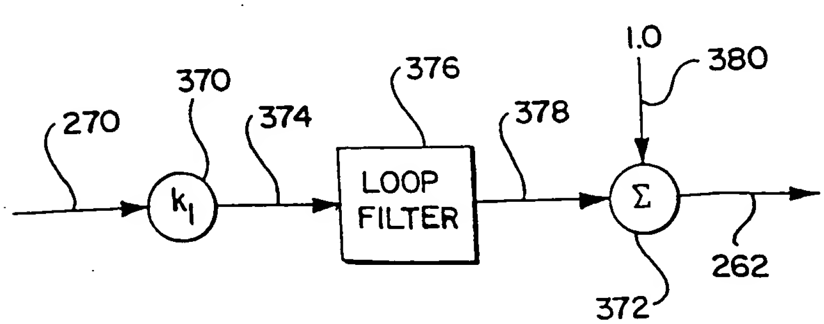

350
332
334
UNIT DELAY
332
358
362
360
364
366
352
UNIT DELAY
+
-
Σ
340
324
354
Σ
336
356
340
338

FIG. 15


1.0
380
370
376
270
374
378
k_1
LOOP FILTER
Σ
262
372

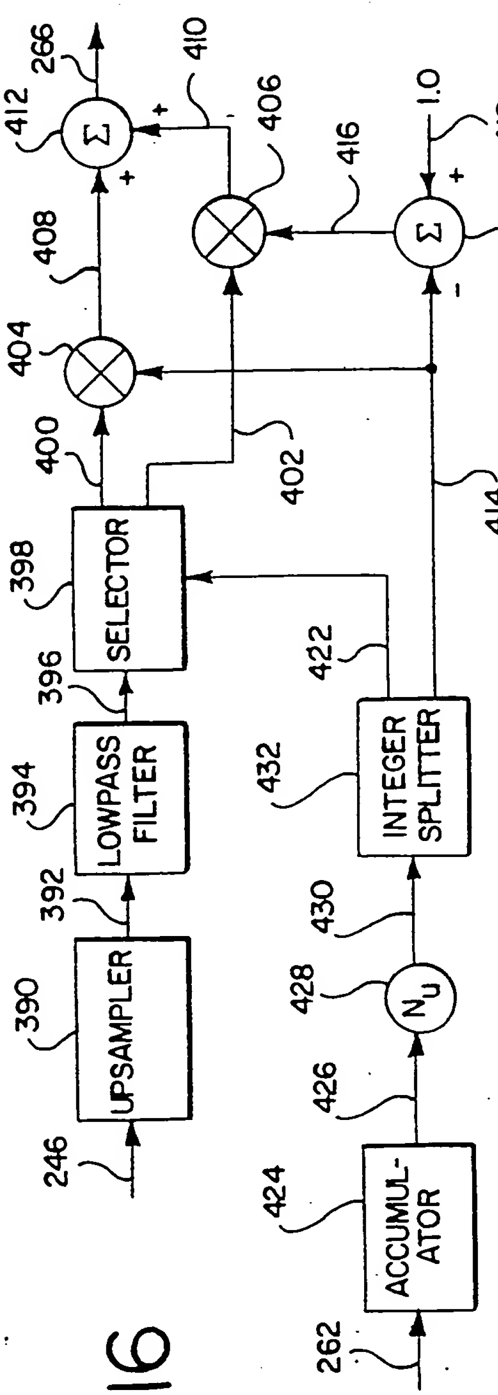

UPSAMPLER
LOWPASS FILTER
SELECTOR
INTEGER SPLITTER
ACCUMUL-ATOR
N_U
Σ
1.0
246
390
392
394
396
398
400
402
404
406
408
410
412
414
416
418
420
422
424
426
428
430
432
262
266

FIG. 16

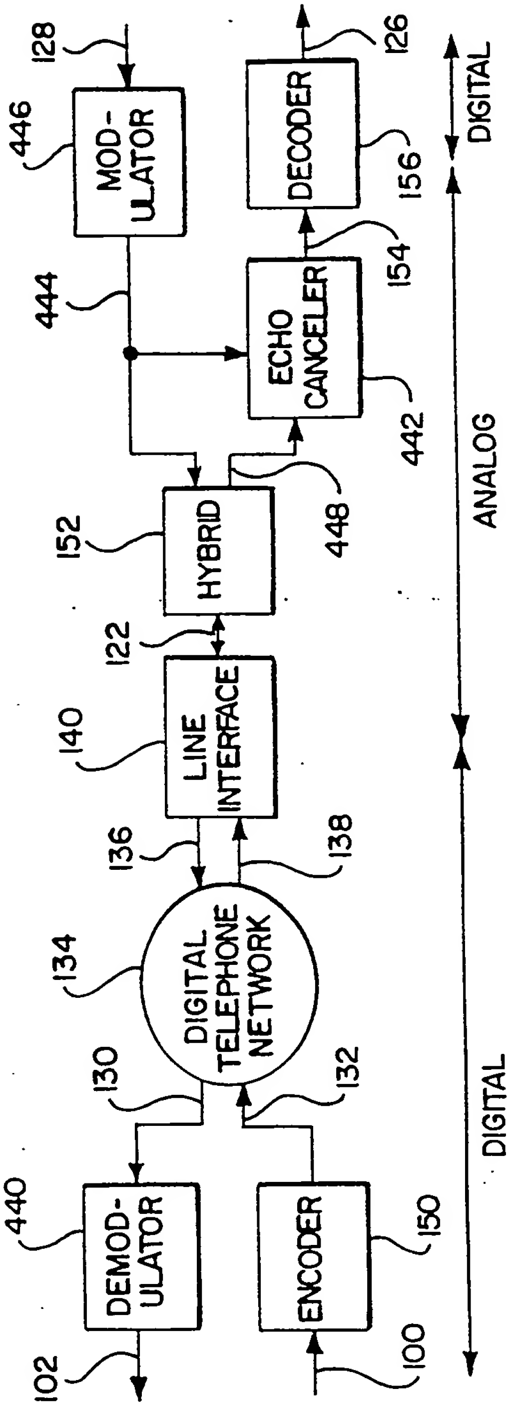

MOD-ULATOR
DECODER
ECHO CANCELER
HYBRID
LINE INTERFACE
DIGITAL TELEPHONE NETWORK
DEMOD-ULATOR
ENCODER
DIGITAL
ANALOG
DIGITAL
128
126
446
156
154
444
442
448
152
122
140
136
138
134
130
132
440
150
102
100

FIG. 17

FIG. 18

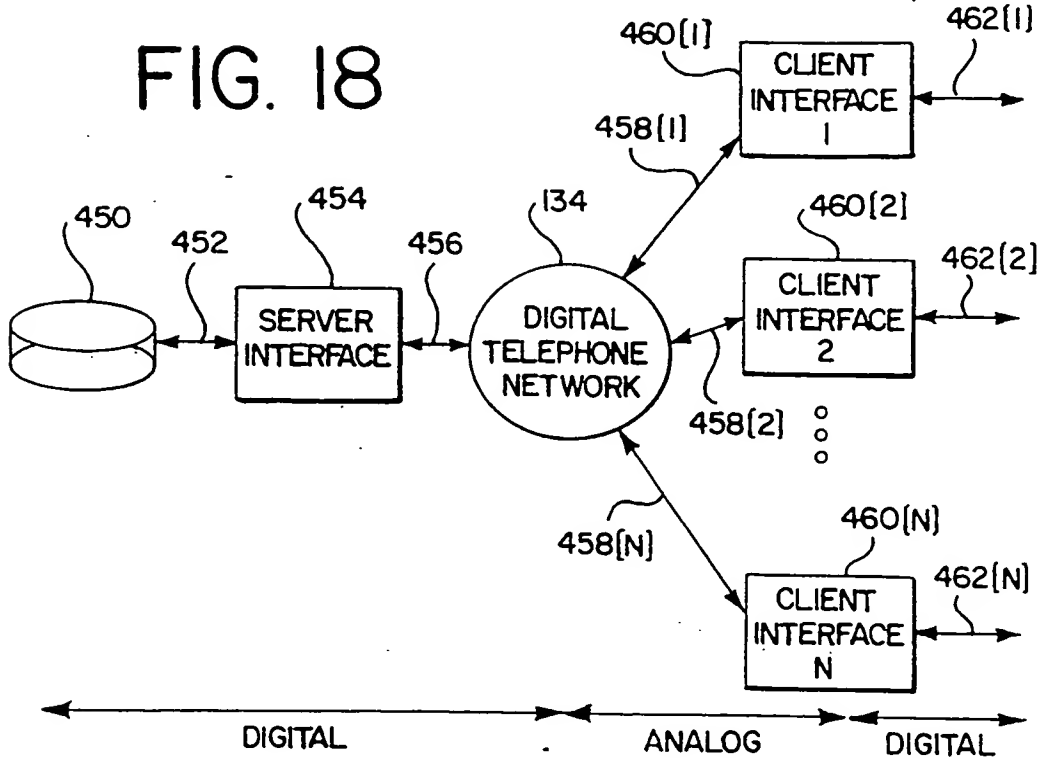


FIG. 19

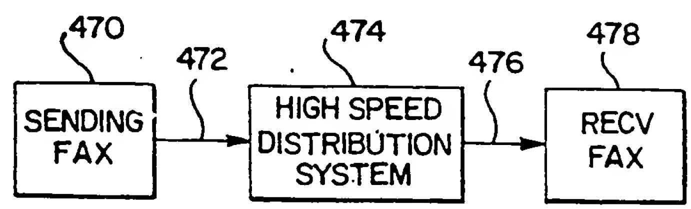
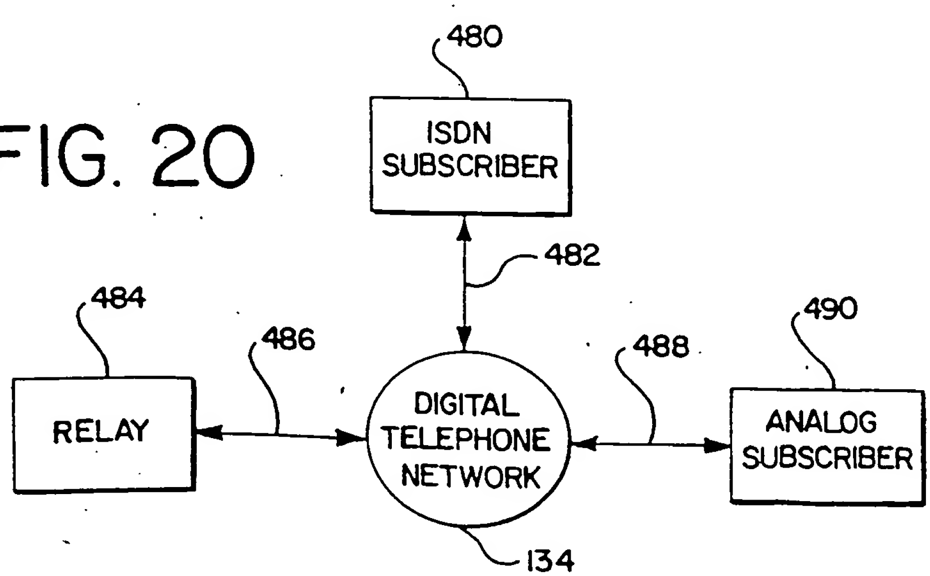


FIG. 20



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		